III 16.367

CIPRIAN-DANIEL NEAGU

DANIELA NEAGU

SISTEME MULTIMEDIA

SERVICII INTERNET

MOTOARE DE CĂUTARE

POȘTĂ ELECTRONICĂ

HYPERTEXT

STANDARDE HTML

PAGINI WEB

CULORI ȘI FONTURI

GIF ANIMAT

WEB DESIGN

APLICAȚII JAVASCRIPT

BAZE DE DATE WEB

APPLET-URI JAVA

Hypermedia

MATRIX ROM
BUCUREȘTI

2. NOȚIUNI GENERALE DESPRE INTERNET

2.1. Servicii Internet

Ce înseamnă să fii pe Internet? O mașină (de calcul) este pe Internet dacă folosește stiva de protocoale TCP/IP, are o adresă IP și are posibilitatea să trimită și să recepționeze pachete IP către/de la celelalte mașini conectate la Internet. Toate acronimele prezente în această definiție sunt explicate în cele ce urmează.

Observație: simpla posibilitate de a trimite/recepționa poșta electronică nu este suficientă pentru a defini un sistem legat la Internet.

Observație: mulți posesori de calculatoare personale pot să apeleze la un furnizor de servicii Internet (ISP - Internet Service Provider) și, folosind un MODEM, să primească o adresă IP temporară, să trimită și să primească pachete IP.

Observație: o mașină face parte din Internet atât timp cât este conectată la router-ul ISP.

2.1.1. Accesul la informație de la distanță

Accesul informației la distanță poate avea diferite scopuri:

- acces la instituții financiare;
- cumpărături electronice folosind cataloage online;
- accesul www (World Web Web);
- accesul la biblioteci digitale de date;
- comunicații interpersonale, extensie a telefonului clasic, se realizează prin întâlniri virtuale: videoconferințe, chat, poștă electronică;
- divertismentul interactiv: este o industrie având ca punct de plecare serviciile video la cerere și ca scop programarea succesiunii emisiunilor și realizarea filmelor interactive.

Accesul la informație de la distanță are ca suport existența rețelelor de calculatoare. Rețelele de calculatoare sunt colecții de calculatoare interconectate (două calculatoare interconectate sunt capabile să schimbe informații). Internet reprezintă cea mai extinsă structură de rețele de calculatoare interconectate, cu o topologie virtuală datorită modificării continue a conexiunilor.

În contextul dezvoltării şi extinderii rețelelor de calculatoare, www (**W**orld **W**ide **Web** sau **w3**), apărut în 1989, a devenit suportul arhitectural pentru accesul la documente răspândite pe toate maşinile din Internet, între care există la un moment dat legături. Informații despre consorțiul www pot fi găsite la adresa `http://www.w3.org`.

2.1.2. Aspecte privind utilizatorul www

Din punctul de vedere al unui utilizator, w3 constă într-o colecție imensă de documente. Documentele sunt pagini Web răspândite în toată lumea, pe calculatoare conectate la Internet. Fiecare pagină poate conține legături (link) către alte pagini aflate oriunde în lume.

Despre paginile care indică spre alte pagini, se spune că utilizează **hypertext.** (text virtual). Paginile pot fi vizualizate cu ajutorul programelor de navigare (browsere), cum ar fi Mosaic, Netscape, Internet Explorer. Un browser aduce pagina cerută de pe server-ul gazdă pe calculatorul utilizatorului, interpretează hypertextul şi cozile de formatare şi afişează pagina pe ecran.

2.1.3. Aspecte privind server-ul www: protocoale

Pentru orice server w3, un anumit protocol urmăreşte port-ul 80 TCP (Transfer Control Protocol) pentru conexiuni inițiate de către clienți: După stabilirea unei legături, clientul trimite cererea de adresă Internet, iar serverul trimite raspunsul, după care eliberează conexiunea. Protocolul reprezintă un set de reguli ce descriu modul în care două calculatoare pot comunica prin intermediul unui modem sau al unei rețele. Protocolul care descrie cererile şi răspunsurile permise pe Internet pentru transferul informației hypertext între serverul şi clientul www se numeşte HTTP: Hyper Text Transfer Protocol.

Un alt protocol utilizat frecvent este FTP: File Transfer Protocol. Acesta reprezintă cel mai răspândit mod de transfer al fişierelor între calculatoarele legate la Internet. Prin FTP, universitățile, serviciile guvernamentale, agențiile, companiile naționale sau internaționale oferă fişiere ce pot fi copiate pe sistemul utilizatorului de servicii Internet. O mare parte a acestor informații sunt însă protejate prin parole, pentru a restrânge accesul doar la anumite categorii restrânse de utilizatori.

Informațiile cu caracter freeware, însă, sunt disponibile pe Internet prin intermediul transferului **Anonymous FTP**. În acest caz special, utilizatorul semnează în calculatorul gazdă ca având numele "guest" sau "anonymous" în locul numelui său real.

Utilizatorul, în momentul selectării unei legături Internet, face apel implicit către o pagină aflată pe un server specificat. Această

pagină se identifică prin numele locației: URL (Uniform Resurce Locator). URL reprezintă formatul uniform de localizare a resurselor pe Internet.

Să considerăm adresa Web:

`http://www.w3.org/hypertext/www/TheProject.html`

URL se compune din 3 părți (spre):

- numele protocolului (spre exemplu `HTTP://` sau `FTP://`);
- numele calculatorului pe care se găseşte pagina Web. Acest şir de caractere este secvența de după protocol, începe în general cu www şi se termină la primul delimitator /, conținând caractere alfanumerice despărțite prin punct, şi, eventual, portul de acces la serverul Web;
- calea (succesiunea de directoare din rădăcina serverului Web) şi numele fişierului ce conține pagina Web (implicit index sau default cu extensia .html, .htm sau .shtml)

Figura 2-1. Structura URL.

2.1.4. Etape pentru realizarea unei legături Internet

1. Browser-ul determină URL-ul cerut de utilizator;
2. Programul de navigare interoghează DNS (Domain Name Server) care este adresa IP corespunzătoare pentru numele maşinii (spre exemplu, `www.w3.org`);
3. DNS răspunde cu, spre exemplu, `18.23.0.23` (adresa serverului);
4. Programul de navigare realizează o conexiune TCP cu portul 80 al serverului de pe Internet numit `18.23.0.23`;
5. Se trimite de la server-ul clientului o comandă `GET` (de transferare) a fişierului local (spre exemplu, aflat pe server pe calea: `hypertext/www/TheProject.html`);
6. Serverul `18.23.0.23` transmite fişierul `TheProject.html`;
7. Conexiunea `TCP` este eliberată;
8. Programul de navigare (browser-ul) afişează conținutul fişierului de tip hypertext;
9. Browser-ul aduce şi afişeaza toate celelalte fişiere multimedia identificate conform informațiilor din fişierul HTML.

2.2. HTML

HTML (Hyper Text Mark-up Language) reprezintă un limbaj de marcare a hypertextului. În HTML nu se vorbeşte despre programare, ci despre marcarea unui text obişnuit şi de organizarea acestuia. Pentru amănunte, este preferabilă parcurgerea declarației de intentii W3:

`http://www.w3.org/pub/WWW/MarkUp/Activity.`

OBS: Informaticienii care lucrează în HTML nu sunt numiți programatori, ci proiectanți (designers, developpers).

HTML s-a dezvoltat ca o subdiviziune a SGML (Standard Generalized Mark-up Language): un limbaj de nivel înalt de marcare şi descriere a codului, dezvoltat la DOD (Departament of Defense) al USA.

Un browser WEB nu este doar un program de afişare a textului obişnuit, ci interpretează modul în care textul va afişa etichete HTML. După introducerea unui text obişnuit într-un document Web, se setează elementele de marcare care vor anunța browserul cum sunt organizate etichetele documentele html.

2.2.1. Grupul de lucru HTML

Standardul HTML este menținut şi dezvoltat de HTML Working Group, din cadrul organizației w3. Primul standard HTML impus a fost HTML 2.0 (în 1994). Astăzi, versiunea standard este HTML4.0. Acest grup impune standardul de bază pentru formatul HTML şi agrearea standardelor tehnologiilor orientate Web relative la formate grafice, formate de animație, formate sonore, limbaje de programare Web (Java) sau modelare 3D (VRML: Virtual Reality Modelling Language). Orice inițiativă în privința responsabilității dezvoltării Web este aprobată doar de consorțiul W3.

2.2.2. Marcarea unui text

Elementul de bază al unei pagini HTML este textul ASCII. Este prin urmare text obişnuit, deci o pagina HTML poate fi creată cu orice editor de text, salvând-o ca fişier text cu extensii `.htm` sau `.html`). Marcarea textului obişnuit pentru interpretarea de către un browser constă în adăugarea unor anumite comenzi numite **etichete** (**tags**) în respectivul document, pentru a semnala modul de afişare şi comportarea documentului în browser. Etichetele se referă spre exemplu, la evidențierea textului în pagină (alegerea tipului de font), îngroşarea, sublinierea sau inițializarea caracteristicilor de desenare, culoare etc.

În cazul în care informațiile dintr-un document nu pot fi afișate integral într-un singur ecran, browser-ul adaugă implicit caracteristica de scroll (deplasare) la pagina afișată.

2.2.3. Site-ul Web

Site-ul Web poate fi definit ca o colecție de pagini Web, aflată sub controlul unui grup particular de proiectanți și utilizatori Web. În general, un site Web oferă anumite informații despre o anumită organizație. Intrarea (startul) unui site Web este, în general, o pagină denumită `index` sau `default`.

Restul paginilor site-ului Web sunt accesate prin legaturi si prin harta site-ului. Organizarea unui site Web este de foarte mare importanță pentru proiectantul Web. Organizarea site-ului poate urmări spre exemplu gestiunea pe baza unei hărți, soluție ce propune ca orice utilizator să poată accesa oricare parte a site-ului printr-un efort minim de căutare.

2.2.4. Pagina Web

W3 se compune în particular din pagini Web definite ca simple documente HTML: orice document HTML creat de proiectanți este o singură pagină Web, indiferent de cantitatea de informații incluse).

2.2.5. Intranet / Internet

O altă utilizare a HTML și a tehnologiilor Web este realizarea rețelelor interne ale companiilor folosind modelul Internet. Dacă informația în w3 este concepută pentru a fi accesată de utilizatori externi, aceeași tehnologie se aplică pentru utilizatorii interni.

Această abordare permite angajaților să folosească știri interne, anunțurile și formularele impuse de politica firmei în interiorul rețelei proprii. Folosind aceleași browser-e Web, angajații pot accesa datele din rețeaua internă a companiei. În principiu, proiectarea site-urilor Intranet nu este diferită de cea specifică Internet. Singura diferență reală este modalitatea de organizare și tipul de informație oferită consumului extern, respectiv intern.

2.2.6. Hypermedia

Din momentul introducerii browsere-lor grafice, conceptul de hypertext este integrat cu cel de multimedia. Rezultanta este conceptul de hypermedia. Hypertextul (în primul rând hyperlegăturile) a fost extins la hypermedia (fișiere grafice, video etc.).

> *OBS: o legătură hypermedia se identifică prin faptul că fișierul asociat poate să nu fie de extensie .htm (Microsoft) sau html (Unix/Netscape). Legături hypermedia pot fi fișiere* `.pdf`, `.doc`, `.gif`, `.mov`.

Datorită flexibilității protocoalelor Web, astfel de fișiere pot fi trimise de serverul Web către navigatorul utilizatorului la fel de simplu ca un document html.

2.2.7. Formate de bază ale fișierelor multimedia

Tipurile de fișiere pentru reprezentarea și stocarea informației multimedia sunt în continuă creștere și îmbunătățire, datorită exploziei tehnologice în tehnologia informației, IT (Information Tehnology).

Tabelul 2-1. Formate ale fișierelor multimedia.

Formatul de fișier	Tipul fișierului	Extensia
Sunet (format Sun)	Audio	.au
Sunet MS Windows	Audio	.wav
Sunet standard MPEG	Audio	.mpg, mpeg
Voce Sound Blaster	Audio	.voc
Imagini CompuServe	Grafic	.gif
Fișiere grafice MS Windows	Grafic	.bmp
Desene Apple	Grafic	.pict
Animație 3D	VRML	.VRL
Video standard MPEG	Video	.mpg, mpeg
Video pentru MS Windows	Video	.avi
Prezentare multimedia în standard Macromedia	Multimedia	.ocr
Prezentare multimedia în format Microsoft	Multimedia	.ppt
Text ASCII	Text	.txt, .text
Fișiere postscript	Text	.ps
Fișiere Adobe Acrobat	Text	.pdf

2.3. Aplicații de navigare Web

Navigatoarele realizează identificarea și încărcarea unor pagini Web, afișându-le conform standardelor și convențiilor HTML. Cele mai cunoscute și mai utilizate navigatoare Web sunt:

- Browser-ul `MOSAIC`, produs de NCSA (National Center for Super Computing Applications), a apărut în 1993. Are variante pentru MS Windows, MacIntosh și diferite platforme UNIX. A impus prin tipul proiectării o mare parte din standardele HTML. Este freeware (gratis) pentru persoane fizice, organizații academice și organizații nonprofit. Poate fi preluat prin download ftp de la adresa: `ftp://ftp.ncsa.uiuc.edu` sau prin Web de la adresa `http://www.ncsa.uiuc.edu`.

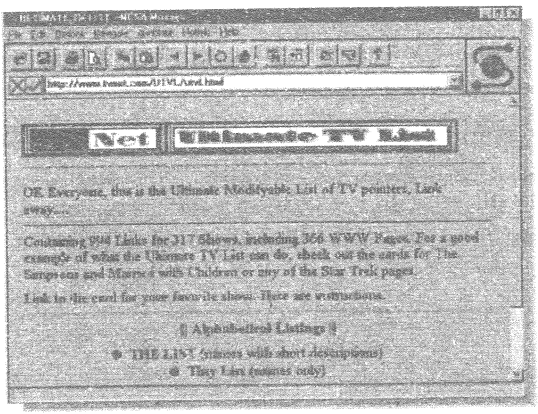


Figura 2-2. Interfața Mosaic.

- `Netscape Navigator` este unul dintre cele mai populare browsere actuale. A apărut în 1995, este total free, fiind proiectat de programatori originali de la NCSA, în cadrul AOL (America On Line). Este considerat un browser Web caracterizat prin flexibilitate şi viteză, şi acceptarea aplicațiilor plug-ins ce extind capabilitățile formelor clasice de browsere pentru interpretarea altor informații multimedia.

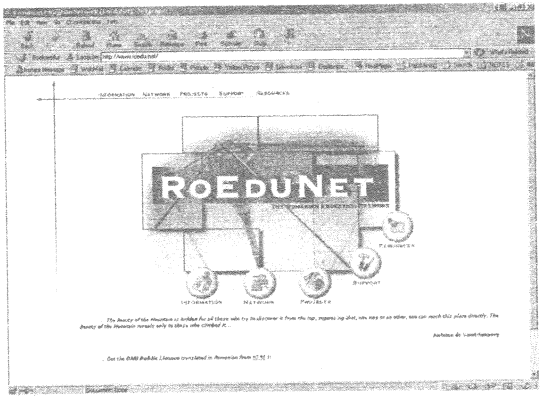


Figura 2-3. Interfața Netscaper Navigator.

- `MS Internet Explorer`, produs de corporația Microsoft, livrat incorporat în pachetele MS Windows. Este freeware şi are caracteristici comparabile cu Netscape Navigator.

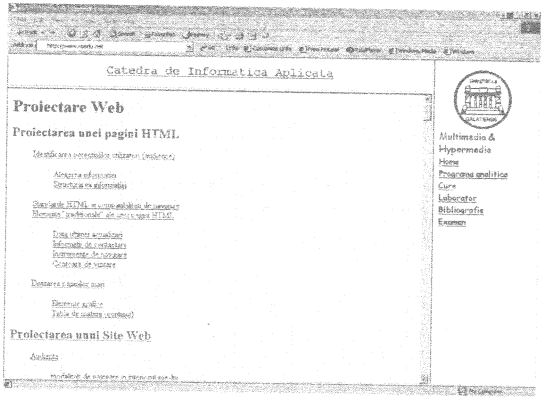


Figura 2-4. Interfața MS Internet Explorer.

Spre exemplu, una din capabilitățile prin care navigatoarele prezintă interfețe user-friendly îmbogățite, o reprezintă posibilitatea gestionării listei personale de adrese Internet (URLs). Bookmarks (Netscape Navigator), respectiv Favorites (Internet Explorer) permit regăsirea site-urilor corespunzătoare printr-un mecanism asemănător celui de completare şi consultare a agendei personale. Adresele pot fi organizate folosind structuri de directoare cu nume personalizate.

În afara textului obişnuit şi a hypertextului, paginile Web pot să conțină: pictograme (icon-uri), desene, hărți, fotografii. Fiecare dintre acestea pot reprezenta eventuale legături spre alte pagini.

> *Observație: nu toate hyperlegăturile indică spre pagini afişabile. Există pagini ce conțin înregistrări audio sau (şi) video, în condițiile în care nu toate browser-ele reuşesc să afişeze pagini hypermedia.*

> *Observație: în cazul paginilor Web cu imagini de dimensiuni mari (a căror încărcare durează mult timp), browser-ele tratează încărcarea lentă afişând întâi textul, apoi imaginile.*

> *Observație: pentru a permite utilizarea unui program de navigare Internet este necesar un calculator cu conexiune directă Internet, sau cu o conexiune la un router sau la un alt calculator legat direct la Internet.*

2.4. URL – Uniform Resurce Locator

Cele mai multe calculatoare legate la Internet sunt identificate prin servicii ce folosesc adresele repartizate acestor calculatoare. World Wide Web a impus propria schemă de adrese, care are forma următoare:

`<protocol>://[//]<host.domain.first_level_domain>/[<path>/<file_name.ext>]`

Observație: <>- semnalizează faptul că șirul interior, închis între paranteze, se înlocuiește cu o secvență reală.

Observație: parantezele [] au sensul semnalizării prezenței opționale a unui atribut.

Exemple:

- `http://www.roedu.net` (Romanian EDUcation NETwork, Rețeaua Română pentru educație)

`http://www.microsoft.com/windows/index.html`, pagină a concernului Microsoft referitoare la sistemul de operare Windows.

Există două avantaje fundamentale ale URL:

- Permite indicarea tipului serviciului Internet dorit (http este serviciul hypertext transfer protocol folosit pentru transferul documentelor Web), dar lista conține protocoalele ftp, file, news, mailto;
- Sistemul URL de adresare realizează obținerea fiecărui fișier, program sau document localizat pe Internet ca entitate adresabilă separat.

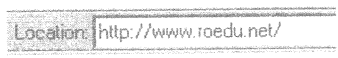


Figura 2-5. Secțiunea `Location` *a Netscape Navigator.*

Figura 2-6. Secțiunea `Address` *a MS Internet Explorer.*

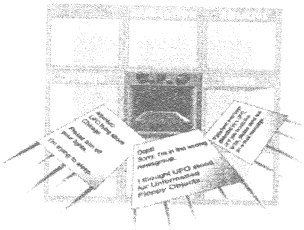
Observație: locația unui document aflat pe Internet se introduce în secțiunea (textbox) numită `Go to` *(*`Location`*, în cazul Netscape) sau* `Address` (pentru Internet Explorer).

2.4.1. Diferite protocoale pentru accesarea informației pe Internet

- `http://` (cel mai utilizat protocol pentru navigarea Internet);
- `https://` (protocol ce asigură câteva opțiuni suplimentare de securitate pentru documentele HTML);
- `file://` (se referă la transferul de documente HTML de pe hard-diskul propriu spre navigator);

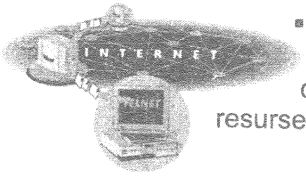
- `ftp://` (file transfer protocol, reprezintă implementarea accesului la servicii de tip ftp. Transferul de fișiere dinspre locația Web spre sistemul utilizatorului se numește **download**, iar în sens invers, de pe sistemul utilizatorului către un server Web se numește **upload**;

- `gopher://` accesează meniuri și documente. Este un sistem anterior World Wide Web, utilizat pentru organizarea și afișarea fișierelor pe servere Internet. servers. Un server Gopher își prezintă conținutul ca o listă structurată ierarhic de fișiere. Datorită expansiunii excepționale a World Wide Web, cele mai multe baze de date Gopher au migrat astăzi pe site-uri Web, deoarece pot fi mai ușor accesate via motoarele de căutare Web. Gopher a fost dezvoltat la University of Minnesota unde a preluat numele mascotei școlii.



- `news://` Permite accesul grupurilor de știri de pe UseNet - parte a Internet care constă în mii de grupuri de discuții, numite **newsgroups**. Usenet permite oricăror utilizatori interesați de aceleași subiecte să comunice între ei;

- `mailto://` (accesează și gestionează mesajele **e-mail**);



- `telnet://` Accesează o sesiune de comunicare directă între două calculatoare, ceea ce permite utilizarea resurselor celui de-al doilea sistem la distanță.

2.4.2. Adrese IP (Internet Protocol)

Fiecare calculator gazdă, respectiv router, din Internet are o adresă IP care codifică numeric adresa de rețea și de gazdă a calculatorului. Combinația numerică este unică, neexistând două mașini cu aceeași adresă IP. Toate adresele au 32 biți. Mașinile conectate la mai multe rețele au adrese diferite în fiecare rețea. Există cinci clase de adrese IP, codificate cu literele A..E (spre exemplu, clasa A conține adresele de la 1.0.0.0 la 127.225.225.225).

Observație: nu toate protocoalele urmăresc aceleași specificații pentru legarea la un URL. Spre exemplu, `mailto://` *este urmat de o adresă e-mail,* `news://` *este urmat de un grup de știri,* `telnet://` *este urmat de adresa sistemului la care dorim conectarea.*

Observație: DNS nu face distincție între majuscule și minuscule;

Observație: fiecare nume de domeniu are cel mult 64 caractere;

Observație: întreaga cale nu trebuie să conțină mai mult de 255 caractere.

2.5. Alte servicii Internet

2.5.1. Poșta electronică

Sistemul de poștă electronică este format din două subsisteme:

- agenții utilizatori, aplicații locale care permit utilizatorului să citească și să trimită scrisori prin poșta electronică (**e-mail**, electronic mail);
- agenții de transfer de mesaje, care transportă mesajele de la sursă la destinatar. Sunt programe de sistem, localizate pe serverele furnizorului de servicii (**ISP**: Internet Service Provider).

Sistemele moderne de poștă electronică fac distincție între plic și conținut.

Plicul (**antetul**) conține adresa destinatarului, prioritatea de transmisie, nivelul de securitate, adresa expeditorului. Plicul conține în general următoarele câmpuri:

- `To`: este adresa e-mail a receptorului primar (poate fi nume de receptor individual sau de grup);
- `CC`: Carbon Copy (copie la indigo) conține lista eventualilor receptori secundari;
- `Subject`: un scurt șir de caractere (opțional) care să caracterizeze conținutul mesajului propriu-zis;
- `Attachments`: unui mesaj i se poate atașa cel puțin un fișier de orice format;

Content (conținutul): reprezintă mesajul propriu-zis.

Formatul adreselor de poștă electronică este:

`<nume_cutie_postală>@<locație>`

unde: `<locație>` este domeniul din care face parte respectiva cutie poștală.

Alte capabilități ale serviciului de poştă electronică

- **Agende electronice** (address book). De obicei, programele pentru e-mail permit utilizarea şi gestionarea unor liste de nume şi adrese electronice ale persoanelor cu care corespondăm frecvent. Avantajul este acela că, în loc să reținem şi să editäm adresa, aceasta este selectată printr-un clic de mouse.

- **Semnături electronice**: conțin informații despre proprietarul cutiei poştale (nume, adresă, telefon, mesaj). Pot fi adăugate la începutul sau la sfârşitul mesajelor.

- **Lista de corespondență** este o listă de adrese de cutii poştale electronice ale persoanelor care doresc să transmită şi să primească mesaje legate de un anumit subiect (similar cluburilor care folosesc corespondența electronică). Odată abonat la o astfel de listă, vom primi copii ale mesajelor trimise de toți membrii listei.

2.5.2. FTP - File Transfer Protocol

FTP este protocolul utilizat la conectarea a două maşini de calcul în vederea transferului de fişiere. Conexiunea este temporară şi serveşte pentru copierea de fişiere: aducerea de fişiere pe sistemul utilizatorului (download) se realizează prin comenzi `get/mget`, iar transmiterea de fisiere către server (upload) se realizează prin comenzi `put/mput`.

> *Observație: numeroase servere ftp acceptă conexiuni anonime ale utilizatorilor, caz în care user_name=anonymous, iar password=adresa_e-mail). Alte servere grupează fişierele dedicate unui subiect anume şi acceptă ftp doar pentru abonații proprii.*
>
> *Observație: pentru evitarea supraîncărcării liniilor pe servere ftp sau http, fişierele principalelor noduri continentale pot fi copiate şi pe alte servere, în general de pe alt continent (**mirror servers**).*

2.6. Limbaje pentru pagini Web

- HTML permite utilizatorului să producă pagini care includ texte, grafică şi legături la alte pagini Web. Pagina Web este interpretată de browser. Un fişier HTML este un fişier ASCII care poate fi editat cu orice editor de texte sau generat folosind generatoare de pagini HTML.

- JAVA, un limbaj de programare orientat pe obiecte, oferă interactivitate paginilor Web. Aplicațiile Java proiectate să ruleze într-o pagină HTML se numesc **applets** şi sunt executate într-un cadru dreptunghiular bine definit în pagina navigatorului Web.

- JavaScript este un limbaj de programare utilizat pentru dinamizarea paginilor HTML, ce rulează pe sistemul utilizatorului, având avantajul/dezavantajul accesibilității on-line a codului.

2.7. Regăsirea informațiilor pe Web

Pentru a facilita identificarea paginilor ce conțin informațiile dorite, aflate pe servere oriunde în lume, au fost scrise programe specifice pentru realizarea indexării Web în diferite moduri. Anumite soluții s-au transformat în aplicații comerciale numite motoare sau mașini de căutare (**search engines**, spiders sau crawlers). Deşi dimensiunile Web sunt practic nelimitate, în esență este considerat un graf imens având în noduri paginile Web, iar ca arce, hyperlegăturile.

Algoritmii de bază se referă, în principiu, la vizitarea tuturor nodurilor unui graf. Dificultatea în acest caz constă în cantitatea imensă de informație de preluat şi în faptul că această informație este în continuă schimbare.

Din punct de vedere al utilizatorului unei mașini de căutare, acesta trebuie să introducă textul căutat şi să acționeze un buton ce porneşte procesul de căutare. Utilizatorul are posibilitatea combinării cuvintelor de căutare folosind operatori; de exemplu, semnul + este folosit pentru AND, iar – pentru NOT. Ca rezultat, utilizatorul va primi adrese de pagini Web unde a fost găsit textul respectiv sau ale căror liste de cuvinte cheie îl conțin, fiind invitat să viziteze pagina respectivă şi să caute eventuale noi cuvinte cheie în acestea.

Cu timpul, site-urile ce oferă motoare de căutare au sistematizat şi catalogat informația cea mai des căutată, propunând o variantă de liste de locații Web predefinită, dedicată anumitor capitole standard. Utilizatorul, în acest caz, trebuie să identifice capitolul şi subcapitolul de care consideră că depinde cuvântul (expresia) de căutare.

Printre serviciile de căutare cele mai cunoscute, identificăm:

- Servere de informații:
  - Excite (`www.excite.com`),
  - Global Network Academy Meta-Library: (`www.gnacademy.org/cgibin/catalogsearch.cgi`)
  - InfoSeek Net Search (`www2.infoseek.com`)
  - Yahoo Search Engine (`www.yahoo.com`)
- Web spiders:
  - Google (`www.google.com`)

- JumpStation II (`js.stir.ac.uk/jsbin/jsii`)
- Northern Light (`www.northernlight.com`)
- WebCrawler (`webcrawler.com`)
- MetaCrawler (`www.metacrawler.com`)

- Motoare de căutare cu interogare în limbaj natural:
  - Ask Jeeves (`www.askjeeves.com`)
- Cataloage:
  - Altavista (`www.altavista.com`)
  - Lycos (`www.lycos.com`)

Excite (`www.excite.com`) este un motor de căutare cu tradiție (lansat la sfârşitul anului 1995), ce funcționează pe principiul web-crawler. În anul 1996 a preluat motorul WebCrawler, combinat cu directorul de la LookSmart. Oferă descrierea a aproximativ 60.000 de site-uri, ceea ce îl califică drept "un director de mărime medie". Are un index declarat de cincizeci de milioane de pagini web.

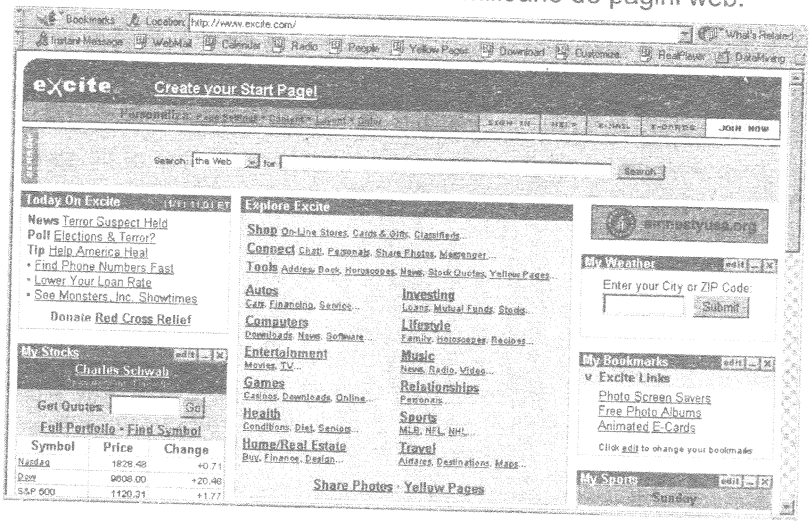


Figura 2-7. Interfața de căutare a serverului Excite.

Excite foloseşte ICE (Intelligent Concept Extraction) pentru a stabili legături între cuvintele cheie indicate pentru căutare şi frazele existente în site-urile pe care le analizează, ceea ce contribuie la contorizarea nivelului de relevanță a site-urilor rezultate din căutare. Pagina de deschidere oferă acces atât la categorii diverse din indexul existent, ceea ce poate simplifica semnificativ căutarea, cât şi la ştiri de ultimă oră, selectate pe web de serviciul News Tracker.

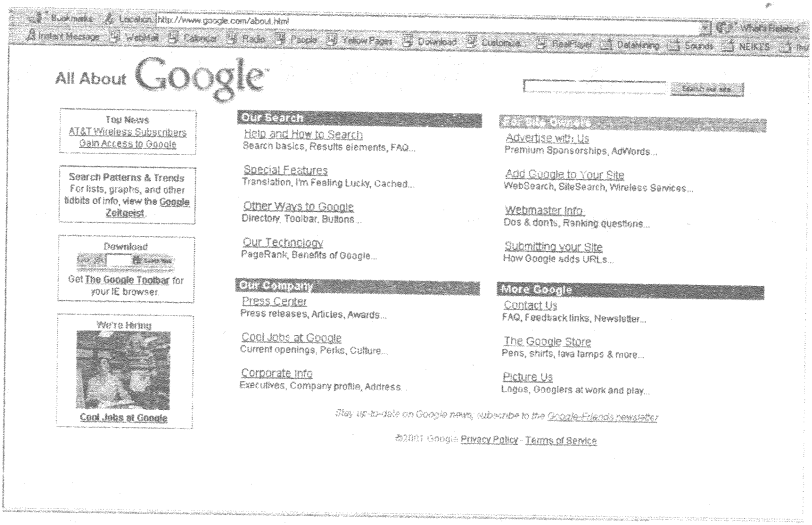


Figura 2-8. Interfața de căutare a motorului Google.

Google este considerat unul dintre cele mai bune motoare de căutare, fiind apreciat pentru nivelul de relevanță al rezultatelor pe care le oferă. O calitate care face Google extrem de apreciat este faptul că face copii ale site-urilor pe care le indexează şi că le păstrează mult timp în cache, oferind astfel un acces alternativ la variante mai vechi ale acestor site-uri. O consecință, cu aplicabilitate rară de altfel, ar fi aceea de salvare a autorilor site-ului respectiv, care pierd informația datorită unor căderi ale serverelor, si a absenței unui back-up (copie de siguranță).

Nivelul de relevanță al rezultatelor obținute la o căutare cu Google provine din faptul că acesta foloseşte, pe lângă celelalte criterii care s-au consacrat deja, o analiză a hyperlegăturilor catre site-uri. Prezența unor porți către un anumit site pe cât mai multe alte pagini reprezintă pentru Google criteriul principal de evaluare a "importanței" informațiilor despre un site de web.

3. NOȚIUNI DE PROIECTARE WEB

3.1. Proiectarea unei pagini HTML

3.1.1. Identificarea potențialilor utilizatori (audiența)

Proiectarea unei pagini Web este o activitate complexă, ce necesită o atenție continuă, pentru care sunt necesare momente distincte în etapa de planificare sau analiză. Analiza paginii trebuie să ia în considerare utilizatorii (vizitatorii) paginilor respective, în sensul înțelegerii modului în care aceştia vor interpreta informațiile din pagină, tipul de calculator pe care îl folosesc, tipul de browser folosit şi viteza conexiunii folosite. Contraexemple în modul de proiectare a paginilor HTML se pot vizita în lista celor mai proaste site-uri Web:

`http://www.Webpractices.com/samplesites.htm`

Identificarea auditoriului este un criteriu crucial pentru proiectarea unui site WEB, ce necesită caracterizarea a cât mai multor informații posibile despre utilizatorii virtuali, incluzând configurația echipamentului (hardware, software) şi conexiunii Internet). Modalitățile de prezentare a informației sunt dependente de:

- nivelul de cultură al categoriei de utilizatori;
- motivațiile navigării pe Web (afaceri, profesionale, personale, divertisment, educationale);
- factorii demografici (vârsta, gradul de educație, localizarea geografică, limba şi limbajul);
- caracteristicile culturale;
- orice alți factori ce pot influența auditoriul în timpul citirii paginii Web.

Proiectul paginii Web trebuie să ia în considerare aceste informații înaintea dezvoltării propriu-zise a paginii, pentru a putea economisi eventualele bucle din procesul de realizare din viitor. În plus, trebuie ca proiectul Web să țină cont de următoarele două caracteristici comune tuturor utilizatorilor:

- aceştia vizitează pagina deoarece sunt virtual interesați de informațiile din pagina respectivă;
- aceştia folosesc pentru accesarea paginii un anumit tip de navigator Web.

3.1.2. Alegerea informației

Atunci când proiectantul alege informația ce va fi publicată în pagină și alege modul de formatare a informației, unul dintre obiectivele principale trebuie să fie găsirea situației optime de minimizare a efortului de citire a paginii de către utilizator. Dacă o pagină are un conținut relevant, prezentat într-o formă bine organizată, cititorul este atras, spre deosebire de comportarea acestuia în cazul în care este "atacat" cu o avalanșă de informații inutile, afișate într-un mod greu de identificat.

Regulile de alegere a informației afișate presupun ghidarea proiectantului după întrebările:

- Ce informație trebuie afișată pentru atingerea obiectivelor de comunicare?
- Ce informație reprezintă subiectul de interes al audienței?

Obiectivele comunicaționale ale proiectantului presupun că acesta are motive corecte și coerente de a publica anumite informații. În acest caz, se caută un conținut abstract al informațiilor și se interpretează mesajele astfel încât rezumatul lor să fie relevant. Astfel, este necesară identificarea unor structuri ale informației publice în etapa de analiză.

Un vizitator al paginii Web are obiective specifice. În mod ideal, acesta dorește să citească toate mesajele afișate la intrarea în site, pentru a deveni conștient de întregul său conținut. Desigur că motivațiile autorului și cele ale cititorului nu sunt întotdeauna convergente! Este necesară includerea unui conținut informativ care să atragă audiența dorită. Trebuie inclusă o cantitate suficientă de informații pentru a atrage, dar nu atât de multă încât să facă mesajul greu de citit.

3.1.3. Structurarea informației publicate

Odata selectată și aleasă informația ce va fi publicată, trebuie gasit modul de afișare a acesteia. Pentru o bună organizare și creativitate, se poate baleia întreaga gamă de etichete standard HTML, ca și cele extinse (caracteristice doar anumitor tipuri de browsere). Aceste modalități de structurare includ:

- **Utilizarea paragrafelor**. Fiecare paragraf se referă la o idee unică, are un singur subiect. Paragrafele se despart în general prin spațierea de paragraful următor (figura 3-1).
- **Formatul caracterelor**. Cuvintele cheie și frazele importante trebuie evidențiate prin alte formate de literă decât cele uzuale: îngroșat (**bold**), înclinat (*italic*), corp de dimensiune fixă (`plain`). Suplimentar, se pot modifica dimensiunea și culoarea textului (figura 3-2).

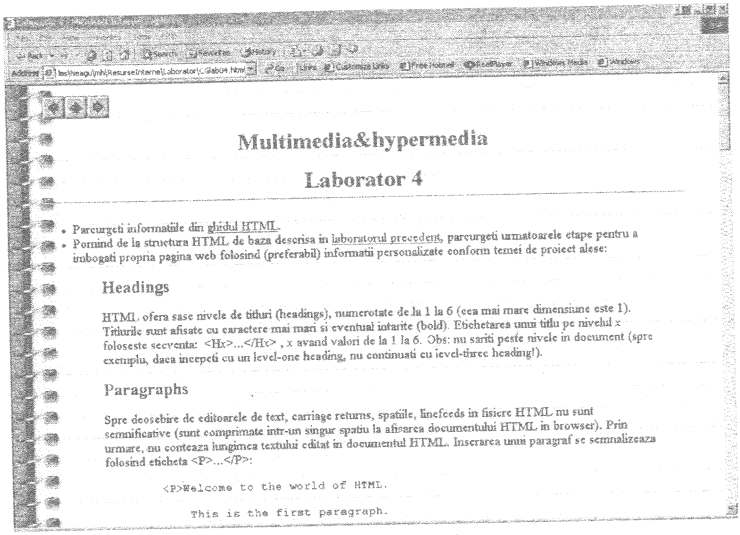


Figura 3-1. Pagină HTML cu informația divizată în paragrafe.

- **Liste HTML**. Listarea permite plasarea articolelor într-un format cu marcare sau numerotare, astfel încât informația să fie prezentată citeț şi uşor de înțeles (figura 3-2). Listele numerotate (ordonate) permit introducerea unei relații de ordine între articole. O listă de definiții, spre deosebire de listele marcate sau numerotate, furnizează atât mijlocul de prezentare a anumitor termeni, cât şi posibilitatea definirii acestora. Spre exemplu, prima pagină poate fi o listă de definiții a cuprinsului paginilor secundare.

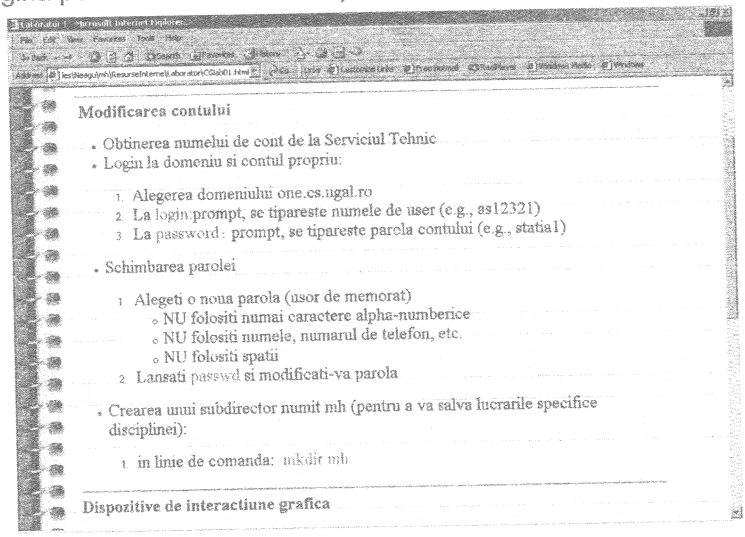


Figura 3-2. Pagină HTML cu informația divizată în liste.

- **Includerea imaginilor**. O imagine, în sens simbolic, ține locul a cel puțin 1.000 de cuvinte: conținutul grafic al paginilor Web este unul din motivele care au făcut Internetul atât de popular! O imagine bine aleasă comunică mesajul repede şi clar. Imaginile pot fi folosite ca: tapet (wallpaper), legături (hyperlink-uri), anexe de navigare (figura 3-3, `http://www.cotnari.ro`).

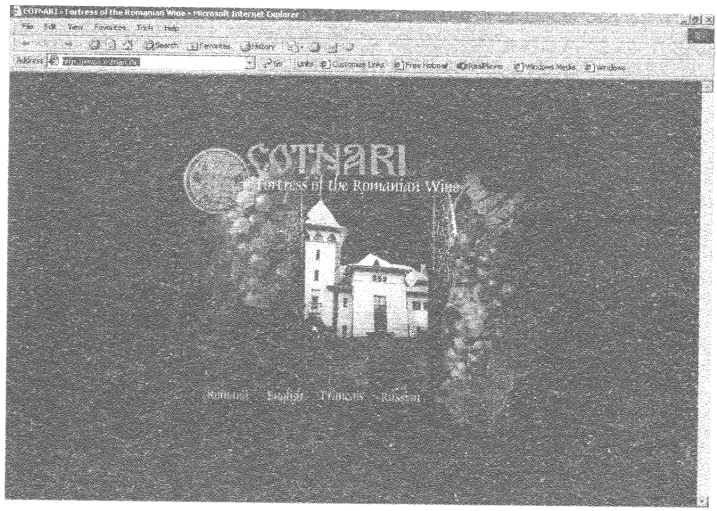


Figura 3-3. Pagină HTML cu informația grafică (imagini).

- **Conținutul multimedia**. Odată cu mărirea vitezei de conectare, compatibilitățile multimedia ale paginii Web au devenit accesibile. Calitatea elementelor multimedia atrage utilizatorii prin combinații grafică 2D/3D, imagini statice şi animate, sunet, video (figura 3-4).

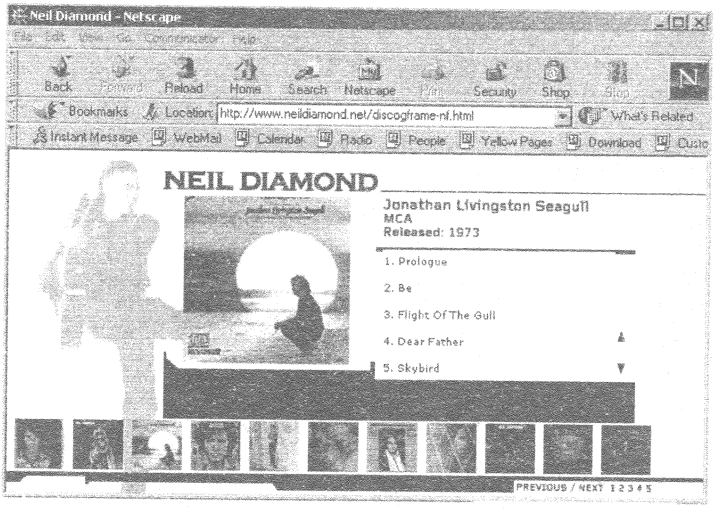


Figura 3-4. Pagină HTML multimedia: sunet şi imagine.

- **Tabele HTML**. Organizarea tabelară a informației permite aranjarea într-o formă uşor de citit. Această formă de organizare ajută proiectantul să controleze alinierea în celule, capabilitate utilizată cu succes pentru a amenaja o pagină placută (figura 3-5).

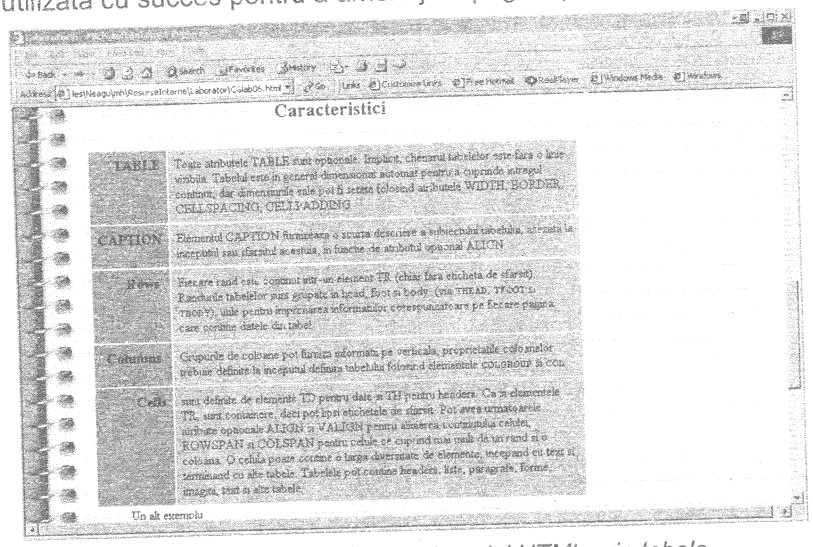


Figura 3-5. Organizarea unei pagini HTML prin tabele.

- **Divizarea în cadre** (frames). Ferestrele de navigare pot fi împărțite în două sau mai multe zone care să includă documente HTML distincte. Astfel de zone distincte ale ferestrei browser-ului se numesc cadre (figura 3-6). Fiecare cadru afişează propriul document, rezultatul fiind publicarea mai multor pagini în fereastră.

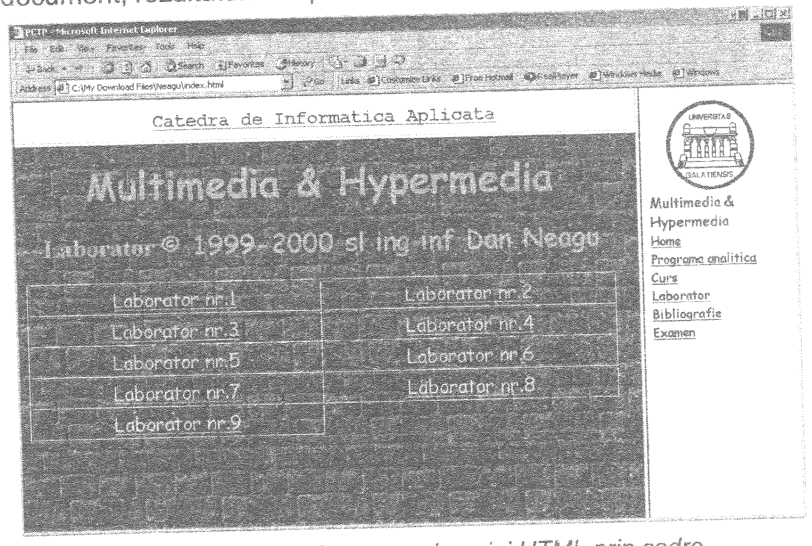


Figura 3-6. Organizarea unei pagini HTML prin cadre.

Observație: cadrele (frames) au fost implementate inițial de Netscape și adoptate relativ recent în Internet Explorer. Există însă (cel puțin teoretic) posibilitatea ca anumite versiuni de browser să nu reușească să citească informația organizată în cadre, afișând eroare.

Aceste elemente ale paginii Web formează o gamă de instrumente de proiectare prin care viitoarea colecție de pagini Web să poată propune auditoriului o informație organizată, ușor de parcurs.

În cazul în care pagina se dezvoltă pentru grupuri neomogene de utilizatori, obiectivul principal devine proiectarea în scopul accesibilității pentru un public foarte larg. În acest caz, ca și în cazul absenței informației despre utilizatorii virtuali, prima regulă ce trebuie respectată devine accesibilitatea maximă a informației.

În cazul în care proiectantul lucrează pentru o pagină de Intranet, atunci beneficiază de avantajele existenței unui grup omogen de utilizatori, ce utilizează, principial, o platformă comună sau cel puțin o colecție cunoscută de latforme (același tip de browser sau din aceeași familie). Aceasta implică proiectarea la nivelul cel mai bun permis de browser. Un alt avantaj îl reprezintă cultura comună: cele mai multe unități ce își dezvoltă structura Intranet au reguli explicite sau/și implicite ce definesc comportamentul pentru situații de lucru. Astfel de reguli vor fi folosite în timpul proiectării.

3.1.4. Standarde HTML și compatibilități de navigare

Instrucțiuni HTML alternative

Una dintre caracteristicile principale ale HTML este aceea că permite furnizarea unui conținut alternativ, în condițiile în care conținutul de bază al paginii nu este afișabil în browser. Spre exemplu, HTML pune la dispoziție instrucțiuni care plasează, pentru o imagine din pagină, un afișaj în mod text care permite să se "vadă" numele fișierului grafic corespunzător.

Pagini HTML alternative

Există situații în care anumiți utilizatori nu pot vizualiza întreaga pagină (cazul cadrelor). Pentru astfel de utilizatori este preferabilă crearea unei versiuni adaptate, spre exemplu fără cadre, cu o pagină de intrare simplă, care să permită acestora să aleagă formatul de vizualizare.

3.1.5. Elemente necesare unei pagini Web

Sunt încetățenite câteva elemente comune ale unei pagini Web, care furnizează utilizatorului informații primare de navigare în pagină:

Ultima actualizare (last update) se referă la cât de recent a fost schimbat conținutul paginii (presupunând că utilizatorul a mai vizitat pagina, această informație îi folosește pentru a se decide dacă merită să o reviziteze, în sensul găsirii a noi informații).

> *Observație: o astfel de informație este preferabil să fie afișată doar pe site-urile modificabile frecvent. Altfel, ar putea să lase o proastă impresie, prin faptul că respectivele pagini nu sunt întreținute.*

Informații de contactare: multe pagini Web specifică posibilitatea de contactare (e-mail) a Webmaster-ului sau a autorului paginii. O astfel de capabilitate reprezintă un feedback între utilizator și autor. În general, adresa de e-mail este o hyperlegătură: figura 3-7, `http://www.upatras.gr` (University of Patras, Grecia).

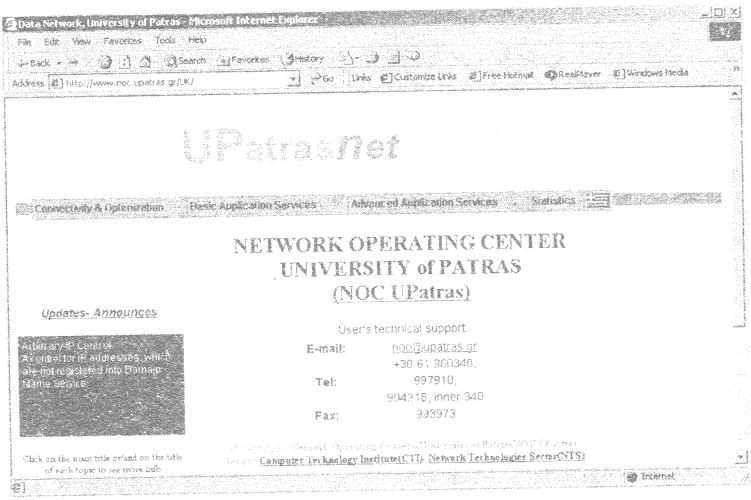


Figura 3-7. Pagină Web care oferă posibilități de feedback.

Instrumente de navigare: se folosesc în cazul în care utilizatorii nu au o viziune clară asupra dispunerii și organizării informației în pagină sau al necesității prezenței legăturilor interactive în paginile de gardă. Pentru aceasta, este imperativ să plasăm instrumente de navigare în pagină (în functie de tipologia utilizatorilor frecvenți). Un vizitator comun al paginii de gardă (home page) se va aștepta să găsească anumite tipuri de imagini, legături, text cu legături sau o hartă în imagini, cu legături către principalele subsecțiuni ale site-ului. O pagină de gardă bine facută va include nu numai o hartă grafică a site-ului, ci și colecția analogă a legaturilor text. Odată accesată o anumită pagină, utilizatorului i se vor oferi bare de navigare sau butoane de navigare către începutul și sfârșitul paginii.

Contorizatoare – reprezintă instrumente de afișare grafică a numărului de utilizatori care au vizitat (figura 3-8) o anumită pagină. Ele pot fi folositoare pentru utilizatorii care doresc să afle câte persoane sunt interesate de conținutul site-ului respectiv, dar mai ales pentru autorii respectivelor pagini, care-şi pot face astfel o idee despre traficul de pagină.

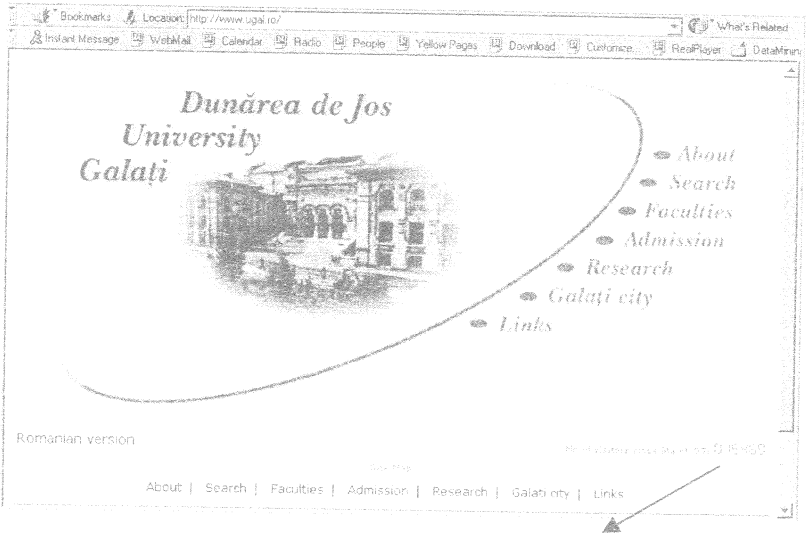
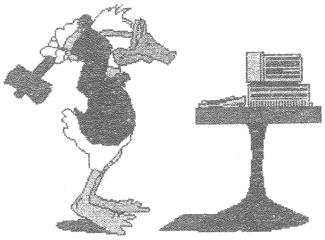


Figura 3-8. Pagină de intrare în web site prevăzută cu contor grafic.

Există două modalități de plasare a unui contor în pagină. Prima abordare implică programarea contorului însuşi. Este cea mai directă cale, dar necesită ca server-ul Web să suporte programe CGI (Common Gateway Interface). A doua posibiltate este utilizată în cazul în care nu există capabilitate CGI, caz în care poate fi folosit unul din serviciile de contorizare on-line. Un astfel de serviciu de contorizare oferă, spre exemplu, `www.digits.com`. În acest caz se permite ca, trecând printr-un proces de setare caracteristic furnizorului de servicii, să se pună la dispoziția autorului de site codul HTML ce trebuie inserat în fișierul HTML pentru introducerea contorului.

> *Observație: în cazul în care se folosesc servere on-line, se va produce o întârziere în transferul paginii catre utilizator, deoarece afişarea contorului va fi livrată de furnizorul la servicii care păstrează valoarea contorului, şi de al cărui server este executată comanda de calcul.*

3.1.6. Divizarea paginilor mari



Una dintre regulile de publicare a informației pe Internet consideră că este preferabilă să nu se plaseze pe o singură pagină un conținut mare de informații. Utilizatorii ar fi astfel forțați să navigheze prin defilare (scroll) prin cantitatea mare de text, în condițiile în care aceasta nu este neapărat necesară. În astfel de cazuri, autorul trebuie să găsească modalități de împărțire uniformă a mesajului în mai multe pagini, astfel încât utilizatorul să o poată citi în secțiuni mici şi uşor de urmărit. În cazul în care pagina nu se poate diviza, este preferabilă utilizarea elementelor grafice propuse în secțiunile următoare, pentru a putea face citirea acestor pagini mari cât mai confortabilă.

Elementele grafice de divizare

Elementele grafice, plasate inteligent, permit crearea unor secțiuni atractive şi interesante:

1. liniile orizontale `<HL>` împart pagina în secțiuni mici (figura 3-9: http://airlab.elet.polimi.it/imagetox), pentru o identificare rapidă.

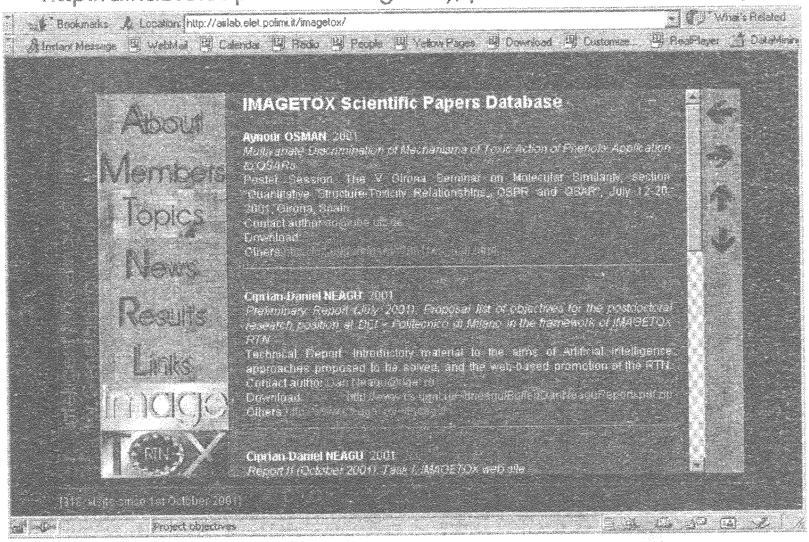


Figura 3-9. Utilizarea liniilor orizontale pentru divizarea informațiilor în pagină.

2. imaginile şi textul se pot completa reciproc (figura 3-10: `http://alexn.dntis.ro`).

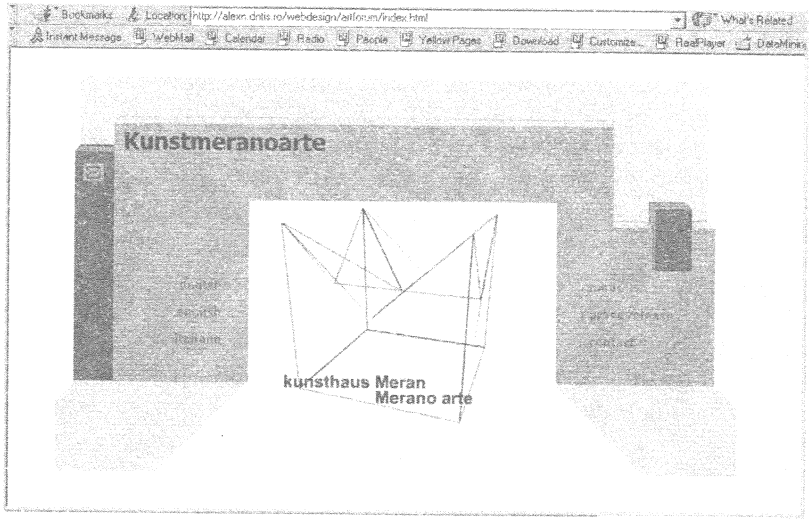


Figura 3-10. Utilizarea imaginilor pentru divizarea informațiilor în secțiuni uşor de urmărit.

3. **Pull quotes** (frazele rezumat) reprezintă fraze cheie din textul propriu-zis, repetate şi evidențiate (fonturi mari, bold, clipitoare), ce subliniază ideile principale şi împart blocuri mari de text (figura 3-11: `http://www.fortressdesign.com`).

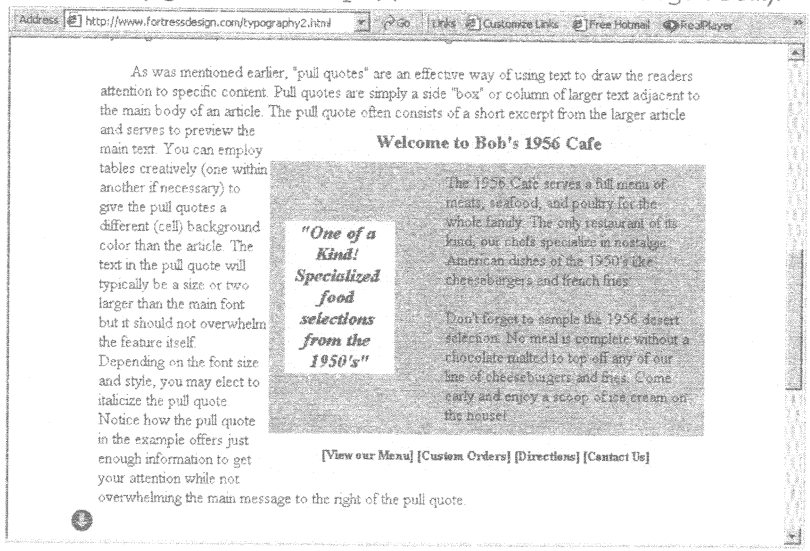


Figura 3-11. Utilizarea frazelor rezumat pentru evidențierea ideilor principale.

Tabla de materii

Dacă o pagină este mare, este preferabil un efort suplimentar pentru a crea o mică tablă de materii la începutul paginii. Aceasta are ca rezultat directarea imediată a utilizatorului către secțiunea din pagină de care este interesat (figura 3-11: `http://www.wdvl.com/Authoring`).

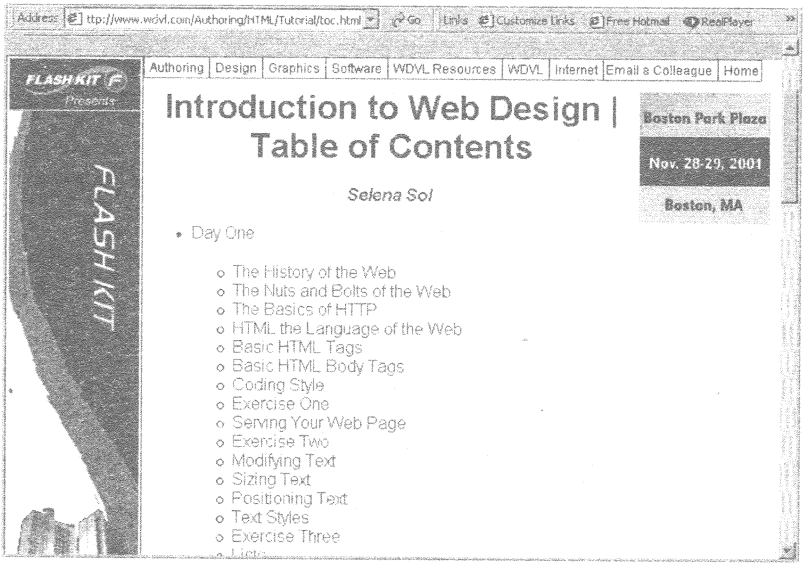


Figura 3-12. Utilizarea tablei de materii pentru simplificarea navigării în pagină.

3.2. Proiectarea unui site Web

Multe dintre subiectele de interes privind proiectarea unei singure pagini Web se regăsesc în proiectarea unui întreg site. Există considerații specifice proiectării de site, deoarece un site Web trebuie să fie mai mult decât o colecție de pagini Web. Site-ul Web devine un instrument de comunicare.

3.2.1. Caracteristicile auditoriului unui site Web

Proiectarea unui site Web trebuie să ia în considerare tipologia auditoriului. Aceasta nu implică neapărat caracteristici ale capabilității de programator a autorului sau calitatea imaginilor, mai ales daca mesajul autorului este difuz. Pentru aceasta, proiectantul trebuie, in timpul procesului de proiectare, să aibă clar conturate tipurile potențialilor utilizatori.

Aceasta depinde practic, pentru un auditoriu larg, de aspectele descrise în următoarele secțiuni.

Cum "navighează" utilizatorul prin informații?

Un site Web diferă de o pagină Web prin structura și cantitatea informației. De aceea, trebuie proiectată o structură cât mai intuitivă, care să permită utilizatorului realizarea unor asocieri naturale. Pentru acest obiectiv, proiectantul trebuie să pună la dispoziția utilizatorului modalități de navigare în interiorul site-lui. Autorul de site-uri trebuie să se folosească de un model general al potențialului consumator, dupa care să se ghideze în timpul procesului de proiectare, în scopul atingerii următoarelor obiective:

- folosirea asocierilor: anumite asocieri sunt general valabile iar altele sunt specifice unor grupuri. Identificând asocierile potențialilor consumatori pentru diferitele articole de informație, le putem exprima în site prin legături hypertext.
- consistența informației: se referă la calitatea grafică, navigarea, prezentarea informației astfel încât utilizatorul să faca un efort minim în scopul creării unei imagini globale a site-ului.

Ce tehnologii folosesc utilizatorii?

Principalul motiv pentru care cele mai multe site-uri Web nu folosesc applet-uri Java sau animații Macromedia se datorează faptului că utilizatorii pot să nu dispună de sistemul, browser-ul și software-ul necesar, ca și de conexiunea necesară. Tehnologiile de care dispun utilizatorii potențiali depind de:

- echipamentul folosit de utilizatori: este un atribut important în etapa de proiectare a unui site Web. Din fericire, HTML este independent de platforma hardware, deci tipul mașinii este în general un aspect irelevant. Totuși, anumite caracteristici pot influența calitatea navigării, cum ar fi colecția de font-uri, algoritmii de decodificare a imaginilor sau prezența aplicațiilor plug-in pentru platforma în cauză.
- monitorul: este preferabilă proiectarea unui site care, sub aspect grafic, să poată fi vizualizat pe monitoare cu performanțe medii (diagonala între 14-17 inches, rezoluția minimă 800 x 600, paleta de minim 256 culori).
- browser-ul Web: proiectul trebuie să cuprindă în general capabilități HTML standard, accesibile majorității programelor de navigare, și nu doar ultimelor versiuni.
- aplicații plug-ins: chiar dacă ultimele versiuni de browser-e sunt din ce în ce mai puternice, conținutul multimedia cu caracteristici speciale (clip-uri audio, video, anumite formate grafice) necesită programe speciale de vizualizare. Astfel de aplicații sunt separate de browser (**aplicații ajutătoare** = helper applications) sau pot

conlucra cu browser-ul (**plug-ins**) pentru afişarea on line a informaţiei provenite din site-ul Web. În aceste cazuri, este preferabil să se verifice dacă utilizatorii virtuali au posibilitatea utilizării unor astfel de aplicaţii şi, eventual, să li se pună la dispoziţie adrese Web de unde pot prelua pentru instalare aplicaţiile respective.

Observaţie: este preferabilă chiar inserarea in pagina de gardă a unor informaţii privind aplicaţiile cu care colecţia de pagini este vizualizată optim.

- viteza de conectare: trebuie luată în considerare cea mai mică rată de transfer a informaţiei către utilizatorii potenţiali. În cazurile pesimiste, este preferabilă micşorarea cantităţii de informaţie multimedia în detrimentul aspectului grafic şi a creşterii informaţiei de tip text, pentru a fi atinse obiectivele pentru care respectivul site a fost proiectat.

3.2.2. Considerarea obiectivelor autorului

Consideraţiile de proiectare orientate către utilizator au o importanţă covârşitoare dar, în nici o fază a proiectării unui site Web, nu trebuie pierdute din vedere motivaţiile autorului. În etapa de planificare a unui site, autorul trebuie să realizeze o listă de obiective din care să rezulte motivele pentru care crează site-ul Web.

Această listă de obiective este un factor important în procesul de proiectare a unui site, prin urmare proiectul site-ului trebuie să fie structurat în primul rând după motivaţiile autorului, considerarea opţiunilor utilizatorului fiind necesară pentru etapa de rafinare, cât şi pentru a ajuta autorul să aleagă modul de comunicare a informaţiei.

3.2.3. Structurarea informaţiei

Caracteristicile auditorului şi obiectivele autorului reprezintă factorii umani ai proiectării unui site Web. Proiectarea unui site depinde şi de alţi doi factori:

- informatia supusă accesului public;
- aspectul grafic al site-ului Web.

În final, trebuie realizat un echilibru în cursul etapelor de proiectare a site-ului între toţi aceşti factori.

Structurarea ierarhică (în adâncime)

Structurarea ierarhică a informaţiei impune organizarea acesteia pe mai multe nivele şi prezentarea în pagina de gardă (home page) a unui meniu cu ajutorul căruia utilizatorul să identifice modalitatea de structurare a informaţiei. Această idee este

similară modului de navigare printr-o structură arborescentă de directoare şi subdirectoare pentru a găsi un anumit fişier. Un exemplu îl constituie site-ul Yahoo (www.yahoo.com), care este structurat (figura 3-13) astfel încât utilizatorul să se poată deplasa cu uşurință de la categorii generale până la informații specifice.

O astfel de abordare furnizează o modalitate sistematică de structurare a conținutului site-ului. Are însă dezavantajul plictisirii şi pierderii subiectului de referință de către utilizator, datorită utilizării prea multor nivele în cazul în care se doreşte deplasarea dintr-o parte în alta a site-ului.

Observație: acest tip de structurare este similar căutarii în adâncime în structuri de reprezentare a informației de tip graf, iar dezavantajul este considerat din punctul de vedere al utilizatorului.

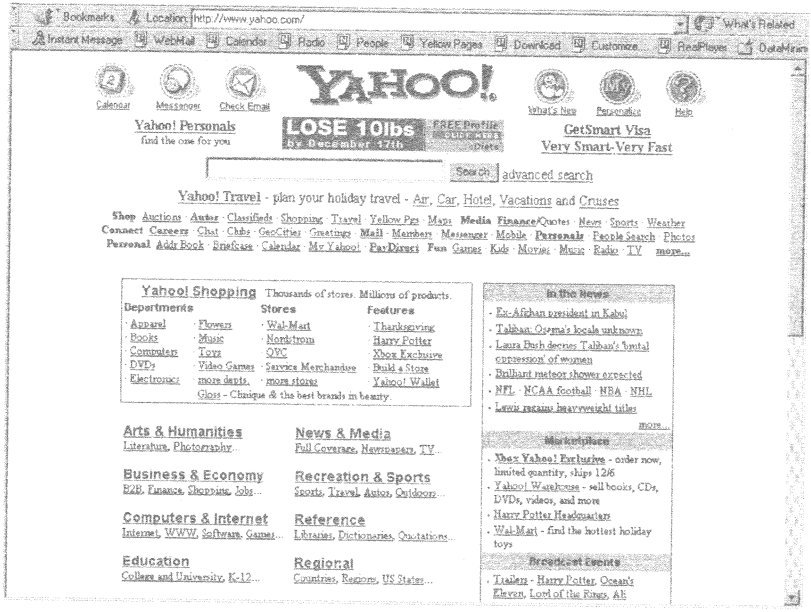


Figura 3-13. Un model de organizare ierarhică a informației.

Unul dintre avantajele prezentării informației în adâncime este faptul că administratorul de site poate interpreta numărul de nivele parcurse de utilizator ca o măsură a interesului acestuia pentru conținutul site-ului.

Structurarea pe orizontală

Orice site are cel puțin unul - două nivele de structurare a informației în adâncime, de la home page către informația detaliată. În scopul micşorării numărului de nivele pe care este

organizată informația și al obținerii unei mai mari accesibilități a informației către utilizator, pot fi folosite următoarele două modalități de liniarizare a structurii informației.

Pentru început, este necesară limitarea numărului de subdirectoare folosite în etapele de început ale proiectării unui site. Proiectantul trebuie să își organizeze structura minimă de directoare pentru gestionarea documentelor HTML.

După această etapă, se poate încerca o liniarizare a structurii informației, prin încercarea de a păstra documentele cât mai aproape de nivelul rădăcinii site-ului. Îmbunătățirea opțiunilor de navigare constau astfel în furnizarea accesului utilizatorilor către cât mai multe secțiuni ale fiecărui nivel.

3.2.4. Dezvoltarea aspectului unui site

De multe ori, aspectul grafic este cel care atrage sau respinge utilizatorul; în plus, un aspect grafic consistent va furniza site-ului o identitate unică. Regula de bază în acest sens este de a furniza mesajul prezentat fără a-l supraîncărca. În consecință, aspectul grafic trebuie orientat către utilizator.

Tehnicile de realizare a aspectului grafic a site-ului Web presupun considerarea încărcării grafice, alegerea colecției de fundal, culoare, a gamei de imagini folosite.

Mai puțin încărcat înseamnă mai mult (Less is often more).

Motivația considerării unui astfel de obiectiv este aceea că, prin posibilitatea afișării mai multor imagini, nu se justifică supralicitarea paginii Web. Dezavantajele tehnice privind acest aspect provin din faptul că unii utilizatori pot folosi browsere de versiuni vechi, iar alții pot avea conexiuni de viteză joasă. Aceste categorii de utilizatori nu vor avea răbdarea să încarce un site bazat pe multă grafică. În consecință, proiectantul trebuie să păstreze grafica (numărul de fișiere) la minim.

În plus, fișierele grafice folosite pentru logo (siglă), reclame (banners) și pentru navigare (buttons) sunt esențiale, dar trebuie uniformizate și alese astfel încât să adauge calitate site-ului prin evidențierea conținutului (figura 3-14, `http://www.jrc.org`).

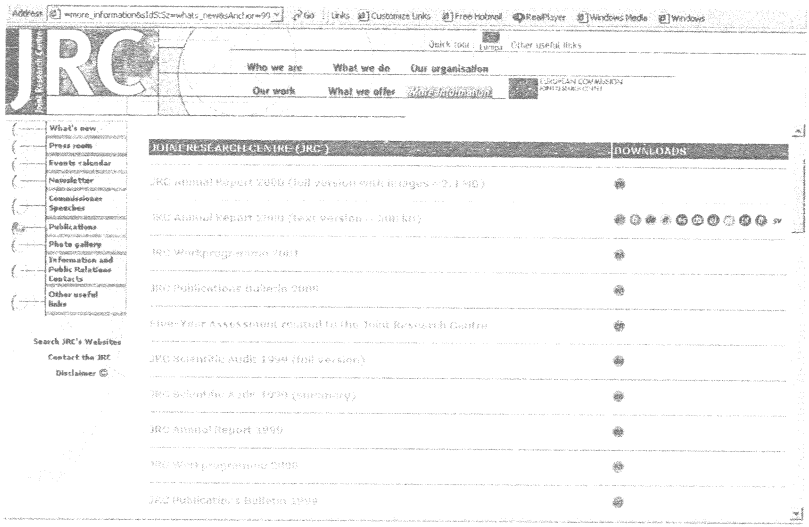


Figura 3-14. Prezentarea grafică a structurii verticale a organizarii unui site Web.

În concluzie, în etapa de proiectare a site-ului, proiectantul trebuie să se decidă asupra setului de imagini folosite şi să continue (figura 3-15) să folosească aceleaşi imagini în toate paginile realizate.

Figura 3-15. Combinarea aspectului fundalului cu informația grafică.

Această strategie îl ajută să promoveze un aspect consistent, cu avantajul că, pentru un utilizator care a trecut măcar o dată recent prin site, fişierele grafice respective se vor regăsi încărcate în memoria "cache" a aplicaţiei de navigare a utilizatorului, prin urmare se va pierde mai puţin timp la o a doua vizită a site-ului.

Fundalul (backgrounds)

Un bun fundal trebuie sa stea exact în fundal: dacă fundalul este prea agresiv, mesajul se pierde, iar utilizatorul va fi influenţat negativ (figura 3-15).

Alegerea culorii

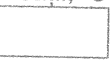
În hypermedia se furnizează posibilitatea controlului culorii şi pentru alte caracteristici ale paginii: fundal, text, tipul legăturii hypertext: nevizitat, vizitat şi activ (figurile 3-14, 15). Culorile acestor trei tipuri de text trebuie alese astfel încât să furnizeze un bun contrast cu fundalul.

Evident, o caracteristică orientată către consumator ar impune ca legăturile vizitate şi nevizitate să fie de culori diferite. Culorile legăturilor hypertext pot reprezenta însă şi o modalitate plăcută de utilizare a culorilor specifice companiei, instituţiei, clubului sau ţării proiectantului (figura 3-16), promovând imaginea respectivă pe întregul site.

Figura 3-16. Îmbinarea culorilor fundalului cu promovarea imaginii apartenenţei proprietarului site-ului web.

Gama de desene (Iconography)

Mulți proiectanți Web folosesc mici imagini pentru a reprezenta secțiunile importante ale unui site: o casa stilizată

pentru legătura Home Page, o căsuță poștală pentru compunerea unui mail; o carte pentru secțiunea de publicații, o tribună pentru secțiunea de prezentări sau un telefon  pentru detaliile privind modalitățile de contactare.

Principalul test al unor astfel de imagini este cel privind cât de intuitive sunt (deoarece, printr-o singură imagine, cât de simplă, se încearcă să se comunice o idee mai complicată, iar utilizatorul să poată să facă o asociere rapidă între imagini şi idei). Cel mai bun mod de testare este folosirea unor utilizatori potențiali, neutri.

3.3. Elementele necesare ale unui site Web

Anticiparea aşteptărilor utilizatorilor reprezintă o caracteristică ce poate atrage şi păstra utilizatorii fideli. Una dintre strategiile clasice este utilizarea elementelor tradiționale ale unui site web, eventual prezentate cât mai modern:

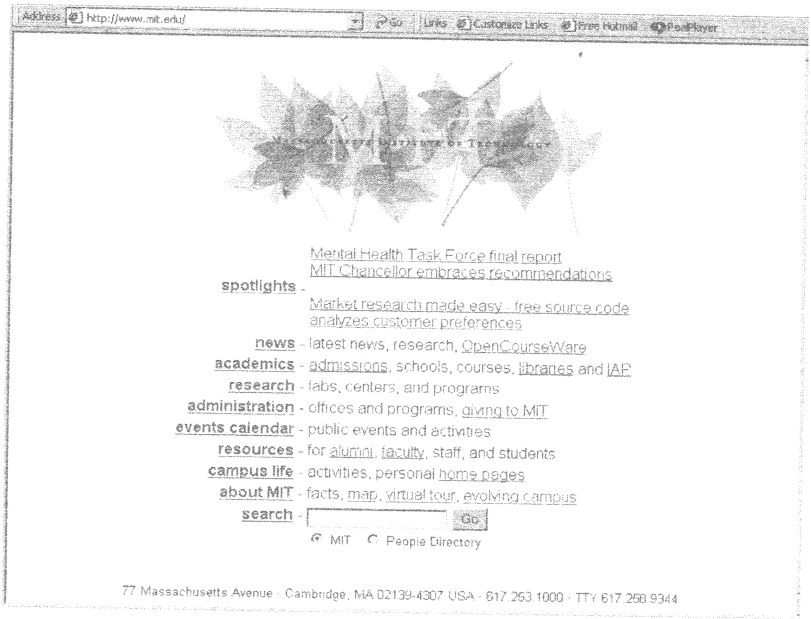


Figura 3-17. Pagina de intrare cu tabelă de materii şi motor de căutare locală.

- **Tabela de materii** (conținutul site-ului, figura 3-17, `http://www.mit.edu`) trebuie să pună la dispoziție legături spre secțiunile site-ului, astfel încât utilizatorul, printr-o parcurgere minimă, să ajungă la informația dorită.

- **Motoare de căutare locale**: indexarea site-ului pentru a-l face mai uşor de parcurs (în sensul de căutare) reprezintă o modalitate de ierarhizare fără a folosi exclusiv structurarea de adâncime (figurile 3-17, 3-18). Pentru aceasta, trebuie utilizată, spre exemplu, o secvență de cuvinte cheie într-un document ascuns, care să păstreze legăturile către paginile specifice. Motorul de căutare propriu devine astfel nimic altceva decât un algoritm de căutare bazată pe text în cât mai puține astfel de documente.

- **Instrumente de navigare** (navigation tools) - trebuie să fie disponibile pe fiecare pagină opțiuni către secțiunile majore ale site-ului (spre exemplu, `back` , `next` , `up` , `down` , `home-page`), eventual şi o listă a tabelei de materii într-un format minimal aflat în "header-ul" sau "footer-ul" paginii HTML, sau eventual lateral folosind cadre. Este preferabilă furnizarea suplimentară a legăturilor către zonele importante ale site-ului (motoare de căutare locală).

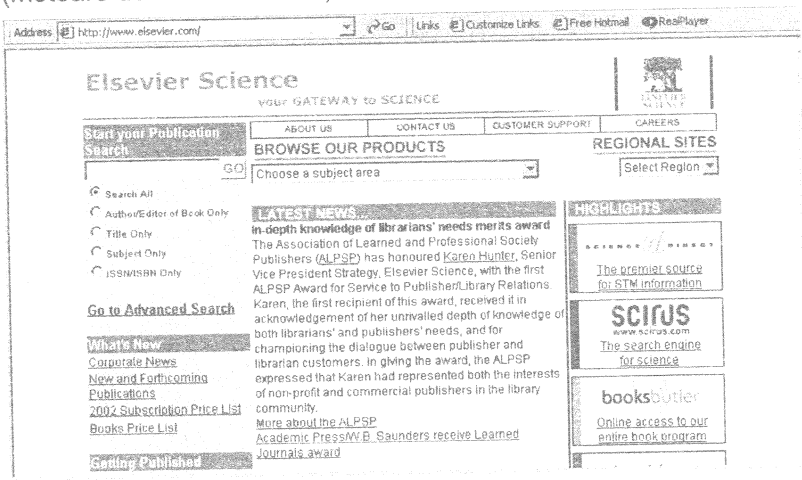
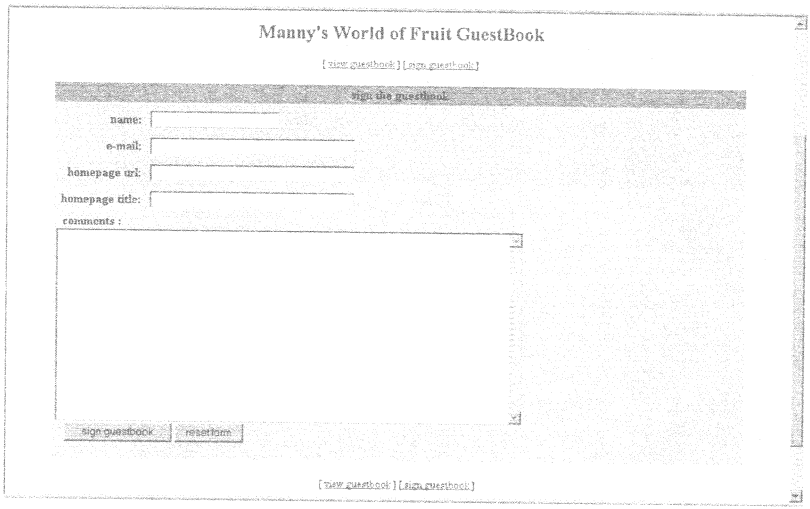


Figura 3-18. Pagină ce evidențiază ultimele noutăți şi secțiunea News.

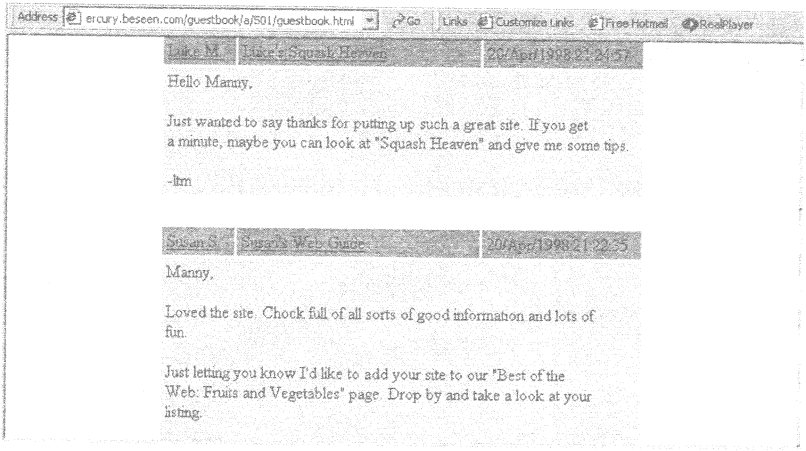
- **Noutăți** (news): utilizatorii frecvenți vor aprecia la justa valoare prezența secțiunii `News` sau `What's new`, deoarece pot fi interesați doar de ultimele modificări. O astfel de abordare va elimina căutarile prin întregul site pentru a identifica eventuale

modificări survenite într-o perioada scurtă de timp (figura 3-18, editura Elsevier, `http://www.elsevier.com`).

Observație: secțiunea noutăților poate fi întreținută static sau poate fi generată prin script-uri (secțiuni de cod) care verifică site-ul după data ultimei modificări şi afişează o listă a fişierelor modificate într-o anumită perioadă de timp. Un astfel de script poate însă uşura munca de căutare dar nu şi calitatea prezentării.



(a)



(b)

Figura 3-19. O pagină a cărții de oaspeți (a) şi lista unor mesaje din cartea de oaspeți(b).

▪ **Cartea de oaspeți** (Guest Book) furnizează posibilitatea înregistrării unei păreri a vizitatorului. O astfel de carte de mesaje personalizate (figura 3-19) foloseşte pagini preformatate HTML pentru a prelua informații de la vizitatori şi a le arhiva pe server, pe baza unui text box (de format minimal) şi a unei aplicații CGI sau ASP.

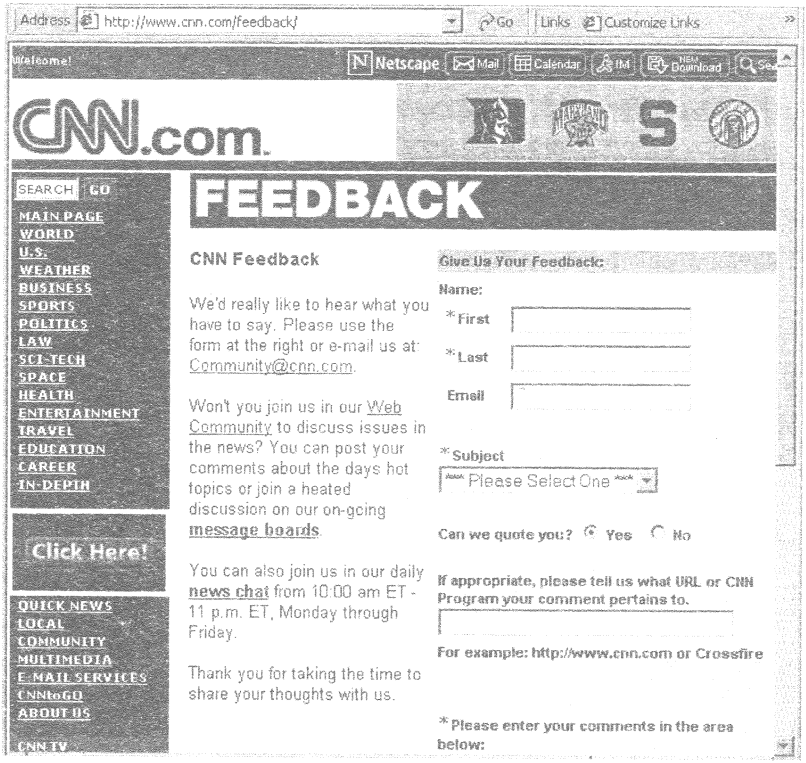


Figura 3-20. Pagină preformatată cu informații de feedback.

▪ **Mecanismul de întoarcere** (Feed-Back) poate avea două forme. Folosirea unei legături hypertext de tipul `mailto:` către o cutie poştală a administratorului site-ului va facilita trimiterea opiniilor utilizatorului către administrator, conducând la un acces direct al informației provenind de la auditoriu. Crearea unei forme HTML (figura 3-20) în care să fie puse întrebări precise prezintă avantajul obținerii unor informații detaliate şi încărcate într-o bază de date, dar numărul şi calitatea informației sunt mult reduse de pierderea de timp luată în considerare de utilizator pentru completarea câmpurilor obligatorii.

▪ **Listele de poştă** (mailing lists) permit utilizatorului să subscrie la o anumita listă de corespondență, prin care să fie anunțați de fiecare modificare a site-ului sau a anumitor subiecte din site.

- **Grupurile de discuții** sunt asemănătoare grupurilor UseNet, dar create pe propriul site. Utilizatorii pot participa la discuții despre site sau despre anumite subiecte caracteristice site-ului, corespondând unii cu alții.

- **Canalele de şuetă** (chat channels) permit utilizatorilor să interacționeze în timp real. Anumite site-uri furnizează canale generale de discuții libere, în care utilizatorii își expun în propoziții scurte părerile; alte site-uri folosesc acest mecanism pentru sesiuni de întrebări şi răspunsuri asupra unui subiect cu un expert sau o celebritate.

- **Conținutul multimedia**: odată cu îmbunătățirea caracteristicilor browser-elor de a afișa în timp real informație multimedia, utilizatorii continuă să se lovească de impedimentul lățimii de bandă, deoarece cele mai multe fișiere multimedia au dimensiuni mari. Pentru reproducerea conținutului multimedia se poate apela la:

  - fișiere audio (în general clipuri prin care, pe un site comercial, se poate face o părere asupra albumului care se cumpără sau pentru un fundal muzical al paginii). Fișierele audio folosite sunt, în general, cele cu extensia `.wav`, `.au`, `.aiff` pentru clipuri de lungime scurtă, respectiv şiruri de sunete şi `.mid`, `.mp3` pentru clipuri muzicale.
  - Fișiere video: `.mpg`, `.gt`, `.avi`. Sunt în general de foarte mari dimensiuni.

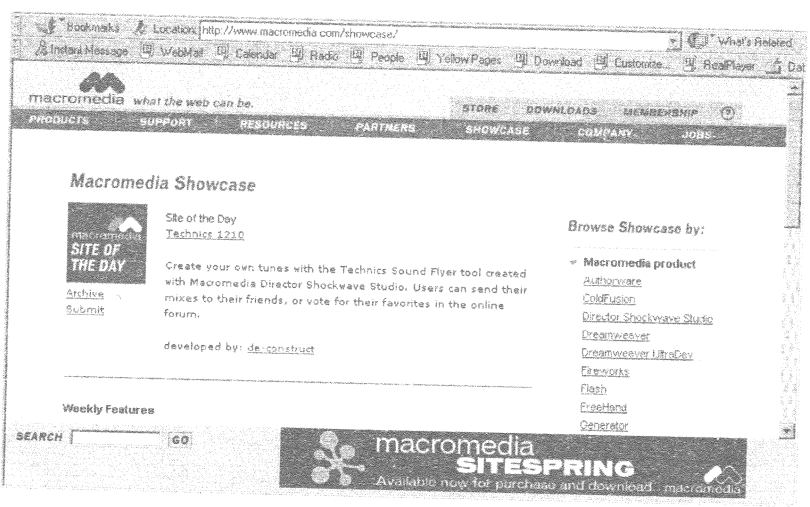


Figura 3-21. Pagina de prezentare a instrumentelor Macromedia.

- Filme şi secvenţe de tip Macromedia Flash/Shockwave sunt folosite pentru prezentări ce combină textul, grafica, secvenţele audio şi video pentru realizarea aplicaţiilor interactive (figura 3-21, `http://www.macromedia.com`).

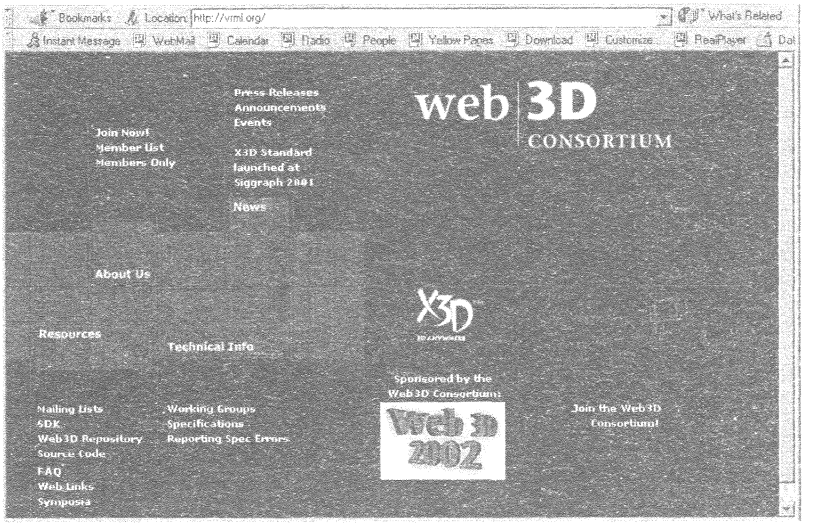


Figura 3-22. Web site-ul consorţiului VRML.

- VRML (Virtual Reality Make-up Language) este o extensie a HTML. Aceasta permite o realizare analogă HTML în spaţiul 3D (figura 3-22). Pentru vizualizare, se folosesc browser-e speciale VRML sau browsere HTML cu un plug-in VRML.

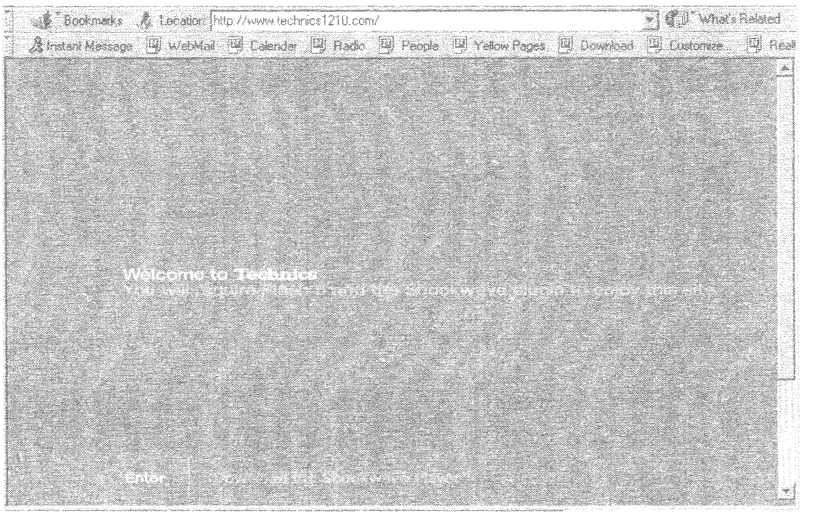


Figura 3-23. Legătură către adresa unei aplicaţii plug-in, în pagini cu atribute speciale.

Observație: cele mai multe fișiere multimedia necesită o aplicație helper sau plug-in pentru a deveni disponibile (în cazul în care utilizatorul nu dispune de un astfel de software, figura 3-23). În principiu, la intrarea într-un astfel de site i se pune la dispoziție adresa Web de pe care să instaleze prin download aplicația.

3.4. Testarea proiectului

Odată terminată activitatea de proiectare, ce are ca rezultat o primă formă funcțională a site-ului Web, trebuie parcursă activitatea de testare a proiectului în scopul determinării posibilităților de îmbunătățire a acestuia, înainte ca site-ul să devină public. Avem trei modalități principale de testare:

- **Testarea pilot** presupune testarea site-ului de câțiva utilizatori potențiali (cu un anume grad de experiență în navigarea Web dar care nu cunosc proiectul în lucru). Aceşti utilizatori trebuie întrebați asupra aspectelor pe care ei le-ar dori modificate (printr-un chestionar cu un set standard de întrebări) şi, eventual, asupra propunerilor.

  Observație: dacă proiectantul nu are la dispoziție astfel de utilizatori neutri, acest tip de testare se poate realiza după ce site-ul Web devine public.

- **Încercarea site-ului** pe diferite browser-e.

  Observație: în general, proiectantul şi-a verificat proiectul în timpul dezvoltării pe un singur browser. Odată finalizat proiectul site-ului, acesta ar trebui testat cu diferite browsere, inclusiv unul negrafic. Eventualele probleme trebuie notate, pentru a conduce la corecții în fişierele sursă HTML.

- **Verificarea site-ului** pe diferite viteze de conectare se poate realiza prin:
  - trimiterea unor note către utilizatori casnici sau de altă natură, în care sunt rugați să acceseze site-ul proiectat (necesită un efort mare în fonduri şi timp). O regulă general acceptată este aceea că o pagina nu trebuie încărcată la utilizator în mai mult de 15 secunde. Paginile care nu îndeplinesc aceste condiții trebuie modificate, în sensul generării mai multor fişiere HTML care să preia segmente ale informației inițiale, şi a micşorării numărului şi dimensiunii capabilităților multimedia pe fiecare pagină.
  - verificarea pe o stație de navigare căreia să i se parametrizeze viteza de navigare.

4. O INTRODUCERE PRACTICĂ ÎN HTML

4.1. Primul document HTML

Pentru primul exemplu de cod hypertext, vom parcurge împreună următoarele etape de creare a unui document HTML, în care vom folosi doar cateva etichete (tags) de bază. Pentru descrierea documentului, vom avea nevoie să lansăm un editor (spre exemplu, joe, în xterm, sub sistemele Linux, sau Notepad sub sistemele MS Windows). Documentul va conține următoarea succesiune de cuvinte:

```
<HTML>
      <HEAD>
            <TITLE>Document title</TITLE>
      </HEAD>
      <BODY>
            Primul document Hypertext!
      </BODY>
</HTML>
```

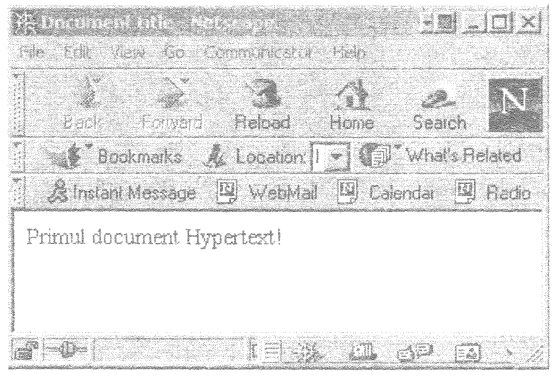


Figura 4-1. Primul document HTML.

Pentru a da o notă personală documentului, putem înlocui Document title cu un titlu personalizat. Să inserăm între etichetele <BODY> şi </BODY> un text propriu (figura 4-1): `Primul document Hypertext!`

Salvați fișierul astfel completat cu extensia `.html` (spre exemplu, `ex1.html` în directorul de lucru).

Vizualizați pagina HTML astfel creată în browser-ul Netscape Navigator sau Internet Explorer.

Deschideti fişierul `ex1.html` folosind un editor de pagini HTML (spre exemplu, Netscape Composer sau MS FrontPage).

Modificaţi atributele de culoare şi aliniere ale textului şi salvaţi fişierul (figura 4-2).

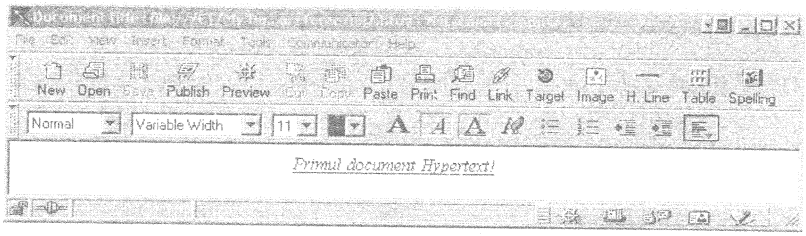


Figura 4-2. Modificarea atributelor textului folosind un generator HTML.

Identificaţi noile etichete inserate de generatorul folosit, vizualizând fişierul cu opţiunea View Source (figura 4-3) sau în editorul text utilizat iniţial. Au apărut practic etichetele `<i></i>` (text înclinat), `<u></u>` (text subliniat), `<center></center>` (paragraph centrat) şi `<font color="#CC0000"></font>` (stabilirea culorii fontului, cu codul RGB în format hexagesimal).

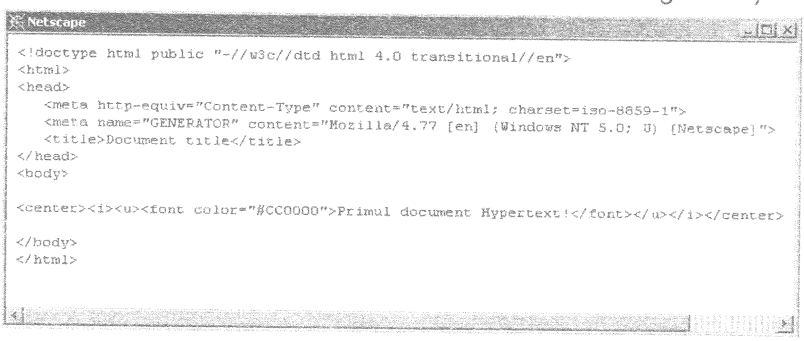


Figura 4-3. Vizualizarea cu opţiunea View Source.

Redeschideţi fişierul astfel modificat (cu opţiunea Refresh/Reload) în browser pentru a vizualiza modificările (figura 4-4).

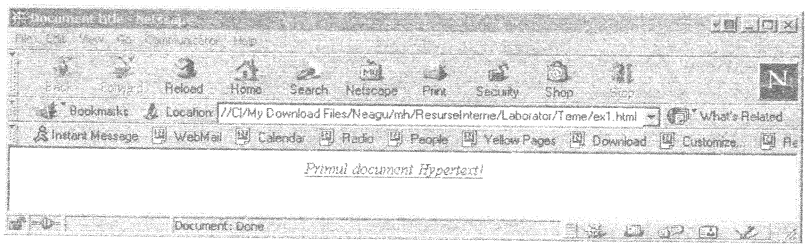


Figura 4-4. Noile atribute ale textului în primul document HTML.

4.2. Titluri şi paragrafe

Pornind de la structura HTML de bază descrisă în secțiunea precedentă, vom parcurge următoarele etape pentru a îmbogăți propria pagină Web folosind (preferabil) informații personalizate.

4.2.1. Titluri

HTML oferă şase nivele de titluri (headings) implicite, numerotate de la 1 la 6 (cea mai mare dimensiune este 1). Titlurile sunt afişate cu caractere mai mari şi eventual întărite (figura 4-5). Etichetarea unui titlu pe nivelul `x` foloseşte secvența: `<Hx>...</Hx>` , `x` având valori de la 1 la 6.

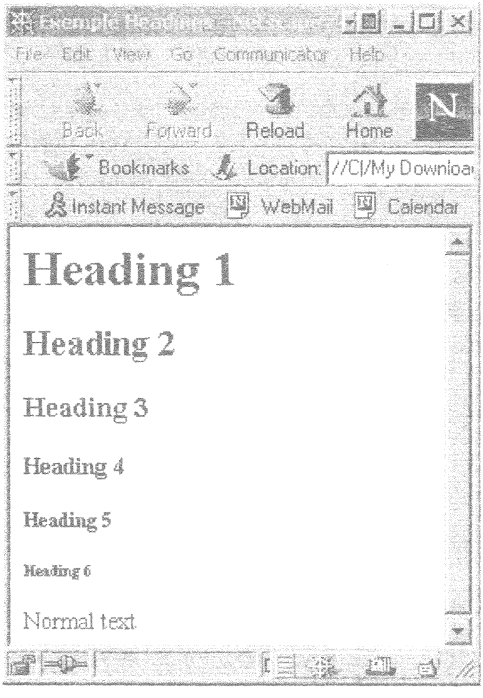


Figura 4-5. Tipurile implicite de titluri în HTML.

Obs: dacă nu doriți să aveți comportări anormale ale textului, nu săriți peste nivele în document! Spre exemplu, dacă începeți cu un titlu `level-one`, nu continuați cu un titlu `level-three`!.

4.2.2. Paragrafe

Spre deosebire de editoarele de text, în fişierele HTML nu sunt semnificative carriage returns, spațiile, trecerile la linie nouă (linefeeds), deoarece sunt comprimate într-un singur spațiu, de către browser, la afişarea documentului HTML. Prin urmare, nu contează lungimea textului editat din documentul HTML.

Inserarea unui paragraf se semnalizează folosind perechea de etichete <P>...</P>:

```
<P>Bine aţi venit în universul HTML!
        Acesta este un paragraf.
        Iar acesta este
        Un text în interiorul aceluiaşi paragraf!
</P>
```

Toate întreruperile de linii sunt ignorate de browser (figura 4-6). Mai mult, fără etichetele de paragraf, întregul document devine un singur şi mare paragraf (cu excepţia textelor preformatate).

Un paragraf poate fi aliniat folosind atributul ALIGN=LEFT/CENTER/RIGHT (opţiunea implicită fiind LEFT):

```
<HTML>
<HEAD>
<TITLE>Initiere in HTML</TITLE>
</HEAD>
<BODY BGCOLOR=BLACK TEXT=WHITE>
<CENTER><H1>Acesta este un Heading 1</H1>
<FONT COLOR=RED SIZE=+3>Welcome to my Page!
</FONT>
<BR>Aceasta este prima mea pagina HTML!
<P ALIGN=CENTEER>ce va fi imbunatatita in curand!
</P>
</BODY>
</HTML>
```

Figura 4-6. Formatarea paragrafelor unui document HTML.

4.2.3. Liste

În HTML sunt propuse etichete pentru utilizarea listelor:

- nenumerotate,
- numerotate,
- de definiții,

inclusiv a celor încuibate (subliste).

Liste nenumerotate (Unnumbered Lists)

Pentru a crea o listă nenumerotată (care folosesc marcatori cunoscuți din tehnoredactare ca bullets):

- începeți prin inserarea etichetei de listă nenumerotată `<UL>`;
- introduceți fiecare articol explicit din listă, precedat de eticheta de articol de listă `<LI>` (**fără** a folosi o etichetă de sfârșit de tipul `</LI>`);
- marcați sfârșitul listei prin eticheta `</UL>`.

Un exemplu de cod pentru o listă nenumerotată, pe care îl puteți insera (figura 4-7) într-unul din fișierele sursă dezvoltate anterior:

```
<UL>
<LI>apples
<LI>bananas
<LI>grapefruit
</UL>
```

Observație: fiecare articol `<LI>` poate conține mai multe paragrafe (marcate prin `<P>...</P>`).

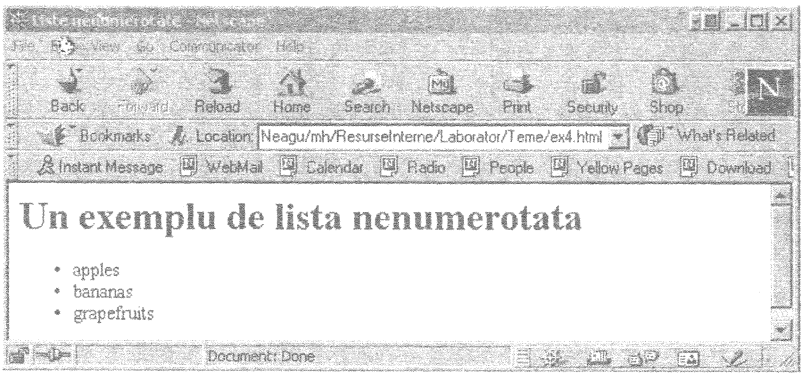


Figura 4-7. O listă nenumerotată.

Liste numerotate (Numbered/Ordered Lists)

O listă numerotată (numită şi listă ordonată) este similară uneia nenumerotată, cu deosebirea că eticheta de listă <UL> este înlocuită de <OL> (articolele listei fiind marcate tot prin eticheta <LI>).

Un exemplu de cod HTML pentru o lista numerotată, pe care îl putem insera într-un document HTML (figura 4-8), aşa cum am procedat în cazul listelor nenumerotate:

```
<OL>
<LI>oranges
<LI>peaches
<LI>grapes
</OL>
```

Figura 4-8. O listă numerotată.

Liste de definiţii (Definition Lists)

O listă de definiţii (marcată prin eticheta <DL>) constă în general dintr-un termen de definit (codificat prin eticheta <DT>, Definition Term), urmat de un enunţ al definiţiei (codificat prin <DD>, Definition Definition). Uzual, un browser Web afisează definiţia pe o nouă linie (figura 4-9), indentată faţă de poziţia termenului definit. Un exemplu de listă de definiţii:

```
<DL>
<DT>NCSA
<DD>NCSA, the National Center for Supercomputing
Applications, is located on the campus of the
University of Illinois at Urbana-Champaign.
<DT>Cornell Theory Center
<DD>CTC is located on the campus of Cornell
University in Ithaca, New York.
</DL>
```

Figura 4-9. O listă de definiții.

Intrările <DT> şi <DD> pot conține mai multe paragrafe sau alte informații referitoare la definiție (inclusiv alte liste). Atributul COMPACT poate fi folosit (figura 4-10) pentru termeni definiți în câteva cuvinte (care se pot găsi pe aceeaşi linie cu termenul definit):

```
<DL COMPACT>
<DT>-i
<DD>invokes NCSA Mosaic for Microsoft Windows
using the initialization file defined in the path
<DT>-k
<DD>invokes NCSA Mosaic for Microsoft Windows in
kiosk mode
</DL>
```

Figura 4-10. O listă compactă de definiții.

Liste încuibate (subliste: Nested Lists)

Listele pot fi încuibate, în sensul că pot conține, pentru unele articole, mai multe paragrafe (figura 4-11), fiecare cuprinzând o sublistă.

Un exemplu de listă cu subliste:

```
<UL>
<LI>A few New England states:
<UL>
<LI>Vermont
<LI>New Hampshire
<LI>Maine
</UL>
<LI>Two Midwestern states:
<UL>
<LI>Michigan
<LI>Indiana
</UL>
</UL>
```

Figura 4-11. Liste încuibate.

Întrerupere de linie (line break)

Eticheta <BR> forțează o întrerupere de linie, în cazul în care dorim să obținem o secvență de linii de dimensiune scurtă în general în cadrul aceluiaşi paragraf.

4.2.4. Linii orizontale

Eticheta <HR> are ca rezultat afişarea unei linii orizontale (Horizontal Rules) de lățimea ferestrei browser-ului. Este în

general folosită pentru a separa secțiunile importante ale documentului. Poate folosi parametri de control (figura 4-12) pentru grosimea (SIZE) şi lățimea (WIDTH - procentul de fereastră acoperit) liniei. Exemplu:

```
<HR SIZE=4 WIDTH="50%" ALIGN="LEFT">
```

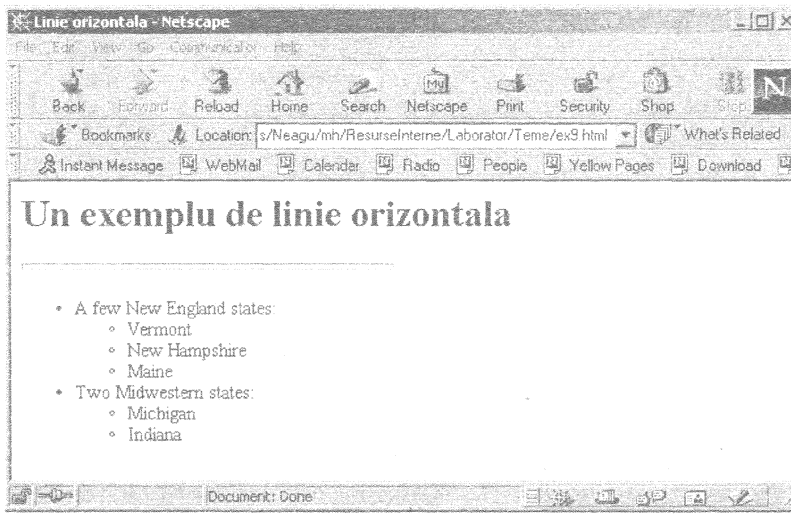


Figura 4-12. Linie orizontală.

4.3. Hypertext

Hypertextul presupune prezența în cadrul colecției de documente a legăturilor între diferite documente sau segmente de document. Legăturile dintr-o pagină HTML permit mutarea dintr-o pagină în alta, astfel încât să putem structura sau referi informații diferite în pagini diferite. Legăturile textuale sunt subliniate implicit: acest mod de semnalizare este considerat implicit, pentru a fi recunoscut textul respectiv ca legătură; legăturile pot fi semnalizate şi prin culori implicite, în funcție de modul de setare a browser-ului utilizatorului şi de formatul folosit de proiectant.

Pentru a formata o legătură pe o linie de text sau pe un cuvânt, se foloseşte o etichetă de forma:

```
<A HREF="DESTINAŢIE">Hypertext</A>
```

În cazul în care legătura indică spre un alt site, adresa acestuia trebuie precedată de o secvență care să indice tipul protocolului utilizat pentru transferul informației (cum ar fi `HTTP://`). Spre exemplu (figura 4-13), un hypertext care să acționeze ca o legătură către site-ul Altavista se realizează cu eticheta:

```
<A  HREF="HTTP://WWW.ALTAVISTA.DIGITAL.COM">Cauta
cu Altavista</A>
```

Figura 4-13. Hypertext.

În locul hypertextului se poate folosi o imagine, ce va constitui simbolul legăturii. Eticheta ce semnalizează închiderea legăturii este `</A>`.

În cazul unei legături către o pagină proprie, aflată în aceeaşi locaţie ca şi documentul în care se găseşte legătura, nu este necesară o legatură care să stabilească protocolul de transfer, ci este necesar doar numele documentului HTML dorit. Spre exemplu, o legătură aflată în documentul `ex10.html` către o altă pagină de informaţii ex9.html aflată în acelaşi folder, va folosi eticheta:

```
<A HREF="ex9.html">Exemplul 9</A>
```

> *Observaţie: pentru a insera o legătură folosind editorul PageComposer, se selectează textul sau imaginea ce se doreşte să reprezinte noua legătură şi se foloseşte butonul Link sau Right click+Insert link.*

Exemplu (figura 4-14):

```
<HTML>
<HEAD><TITLE> My Home Page </TITLE></HEAD>
<BODY BGCOLOR=SILVER>
<CENTER><H1> Welcome to my Page! </H1>
<FONT COLOR=RED SIZE=+3>Hypertext </FONT>
<BR>Un exemplu de legături HTML.
<P ALIGN=LEFT>O pagina anterioara este...
<A HREF="ex9.html">Pagina 9</A>
</P>
<A  HREF="http://www.altavista.digital.com">Cauta
cu Altavista </A>
</BODY>
</HTML>
```

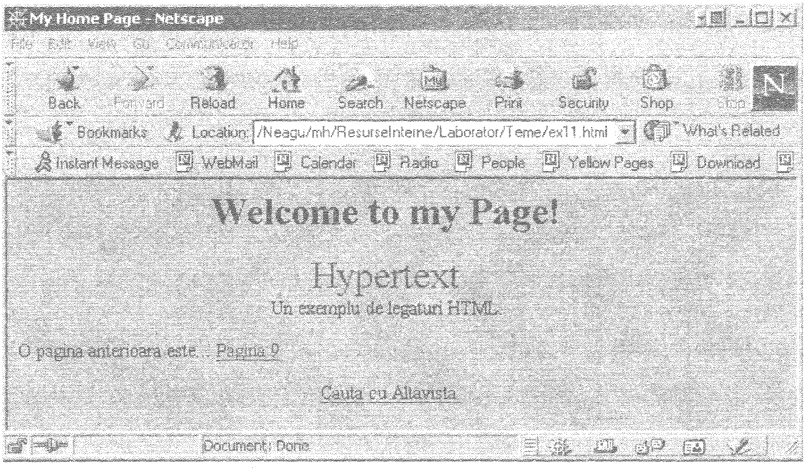


Figura 4-14. Un exemplu de pagină personală hypertext.

Pot exista legături către anumite părți componente ale aceleași pagini, caz în care trebuie inserate ancore (targets) în respectiva pagină. Ancorele sunt introduse prin același tip de etichetă specif hypertextului, iar trimiterea respectă regulile de adresare relativă, numele ancorei urmând numelui paginii și delimitatorului #:

```
<html>
<head>
<title>My Home Page</title>
</head>
<body bgcolor="#C0C0C0">
<a NAME="top"></a><center>
<h1>Welcome to my Page!</h1></center>
<center><font color="#FF0000">
<font size=+3>Hypertext</font></font>
<br>Un exemplu de legaturi HTML.</center>
<p>O pagina anterioara este...
<a href="ex9.html">Pagina 9</a>
<center>
<a  href="http://www.altavista.digital.com">Cauta
cu Altavista</a>
<br><a href="#top">Inceput de pagina</a></center>
</body>
</html>
```

Legăturile relative consideră parcurgerea drumului între documentul cu hyperlegătură și destinație, respctând convenția ca, în cazul în care trebuie să se parcurgă un folder superior ierarhic, fiecare urcare în ierarhie să fie semnalizată de o secvență `../`

4.4. Imagini

Prin adăugarea imaginilor în pagina Web pot creşte calitatea şi aspectul site-ului propriu. Imaginile trebuie să completeze ideile din documentul HTML. Adăugarea imaginilor se realizează prin utilizarea unei etichete de forma:

```
<IMG SRC="IMAGE NAME">
```

unde `IMAGE NAME` trebuie înlocuit cu numele fişierului cu format grafic dorit, inclusiv extensia şi eventual calea către fişier. Spre exemplu, să adăugăm în centrul paginii o imagine de atenționare a faptului că pagina este în construcție (figura 4-15):

```
<HTML>
<HEAD><TITLE>My Home Page</TITLE></HEAD>
<BODY BGCOLOR=BLACK TEXT=WHITE>
<FONT COLOR=RED SIZE=+3>Under construction!
</FONT><CENTER><IMG SRC="gator.GIF"></CENTER>
</BODY>
</HTML>
```

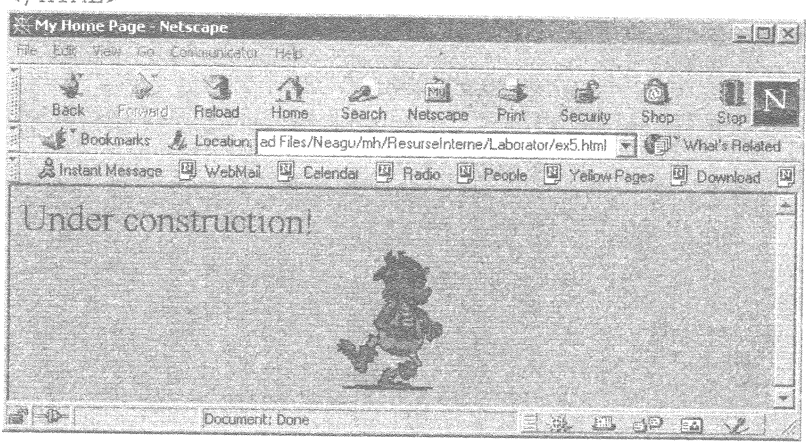


Figura 4-15. Inserarea unei imagini în documentul HTML.

În general se preferă utilizarea unor colecții, eventual personalizate, a imaginilor din site. Astfel de colecții se numesc clipart gallery, şi pot cuprinde:

- Fundaluri (backgrounds);
- Butoane (buttons);
- Imagini animate (animations);
- Bare şi banere de reclamă (bars/banners);
- Simboluri şi pictograme (symbols/icons);
- Alte imagini (ClipArt).

4.5. Tabele HTML

Tabelele permit distribuirea aranjată a datelor (text, imagini, legături, forme, alte tabele etc.) în rânduri şi coloane de celule. Sunt utile în general pentru aranjarea în pagină, deşi tendința actuală este aceea de a folosi stiluri predefinite (style sheets). Folosirea tabelelor poate crea probleme în cazul vizualizării în sisteme ne-grafice.

Tabelele pot conține o primă etichetă optională `caption`, urmată de unul sau mai multe *rânduri* (*rows*). `<tr> ... </tr>`

Fiecare rând este format din una sau mai multe *celule* (*cells*) de tip header `<th>..</th>` sau date (data cells) `<td>..</td>`

Celulele pot fi unite de-a lungul rândurilor sau coloanelor. Acest model furnizează un mod limitat de control asupra formatului paginii (spre exemplu, alinierea orizontală sau verticală a conținutului celulelor, stilului chenarului: borders etc.)

```
<Table border=2 align=right>
<caption>
Un Exemplu
</caption>
<tr>
    <th> a </th> <th> b </th>
</tr>
<tr>
    <th> c </th> <td> d </td>
</tr>
</Table>
```

Rezultatul unui astfel de cod este:

Un Exemplu

a	b
c	d

Parametrii etichetei de definire a tabelelor:

TABLE	Toate atributele `TABLE` sunt opționale. Implicit, chenarul tabelelor este fără o linie vizibilă. Tabelul este în general dimensionat automat pentru a cuprinde întregul continut, dar dimensiunile sale pot fi setate folosind atributele `WIDTH, BORDER, CELLSPACING, CELLPADDING`.
CAPTION	Elementul `CAPTION` furnizează o scurtă descriere a subiectului tabelului, aşezată la începutul sau sfârşitul acestuia, în funcție de atributul opțional `ALIGN`.

ROWS	Fiecare rând este conținut într-un element TR (chiar fără eticheta de sfârșit). Rândurile tabelelor sunt grupate în head, foot și body (via THEAD, TFOOT și TBODY), utile pentru imprimarea informațiilor corespunzătoare pe fiecare pagină care conține datele din tabel.
COLUMNS	Grupurile de coloane pot furniza informații pe verticală; proprietățile coloanelor trebuie definite la începutul tabelului, folosind elementele COLGROUP și COL.
CELLS	Celulele sunt definite de elemente TD pentru date și TH pentru capete de table (headers). Ca și elementele TR, sunt containere, deci pot lipsi etichetele de sfârșit. Pot avea următoarele atribute opționale: ALIGN și VALIGN pentru alinierea conținutului celulei, ROWSPAN și COLSPAN pentru celule ce cuprind mai mult de un rând și o coloană. O celulă poate conține o largă diversitate de elemente, începând cu text și terminând cu alte tabele. Tabelele pot conține header-e, liste, paragrafe, forme, imagini, text și alte tabele.

Un exemplu de tabel cu diferite organizări ale liniilor şi coloanelor, inclusiv reuniri ale acestora (figura 4-16):

```
<HTML>
<HEAD>
   <TITLE>Exemplu de tabel</TITLE>
</HEAD>
<BODY>
<CENTER><H1>Un exemplu de tabel</H1>
<TABLE BORDER=2>
 <CAPTION > Titlul tabelului </CAPTION>
 <TR><TD    ROWSPAN=2></TD><TH    COLSPAN=2>Medie
</TH></TR>
 <TR><TH>Inaltime</TH><TH> Greutate </TH></TR>
 <TR><TD>Barbati</TD>
     <TD ALIGN=CENTER> 69 </TD>
     <TD ALIGN=CENTER> 150 </TD></TR>
 <TR>
     <TD> Femei </TD>
     <TD ALIGN=CENTER> 64 </TD>
     <TD ALIGN=CENTER> 130 </TD></TR>
</TABLE>
</BODY>
</HTML>
```

Figura 4-16. Tabel cu coloane unite.

4.6. Prezentări multi-document: cadre

Cadrele permit afişarea mai multor documente (eventual defilabile) în acelaşi ecran, fiecare caracterizat de propriul său URL. Cadrele pot indica spre documente localizate prin adrese diferite şi pot să fie indicate de alte adrese Internet/Intranet. Documentele vizualizate în acest mod pot constitui ferestre distincte sau subferestre. Această abordare (multiple views) permite proiectanților să păstreze anumite informații vizibile, chiar în cazul în care conținutul altor documente HTML este defilat sau înlocuit.

Spre exemplu, o pagina de gardă poate folosi trei cadre:

- un banner static,
- un menu de navigare,
- un cadru principal, al cărui conținut poate fi defilat sau înlocuit prin alegerea unui articol din cadrul de navigare (spre exemplu, tabla de materii).

O altă situație în care îşi găsesc utilitatea cadrele este aceea în care utilizatorul poate transmite interogări de căutare într-un cadru şi poate recepționa rezultatul în alt cadru.

Un document HTML cu cadre diferă ca aspect de un document HTML obişnuit (fără cadre), deoarece: un document standard are o secțiune `HEAD` şi o secțiune `BODY`, în timp ce un document cu cadre are:

- o secțiune `HEAD`;
- o secțiune `FRAMESET` care defineşte formatul de organizare în fereastra principală a paginii cadru,

- o secțiune opțională `BODY`, care să furnizeze conținutul alternativ al paginii cadru pentru utilizatorii ce folosesc navigatoare ce nu interpretează sau nu sunt configurate să interpreteze documentele organizate în cadre.

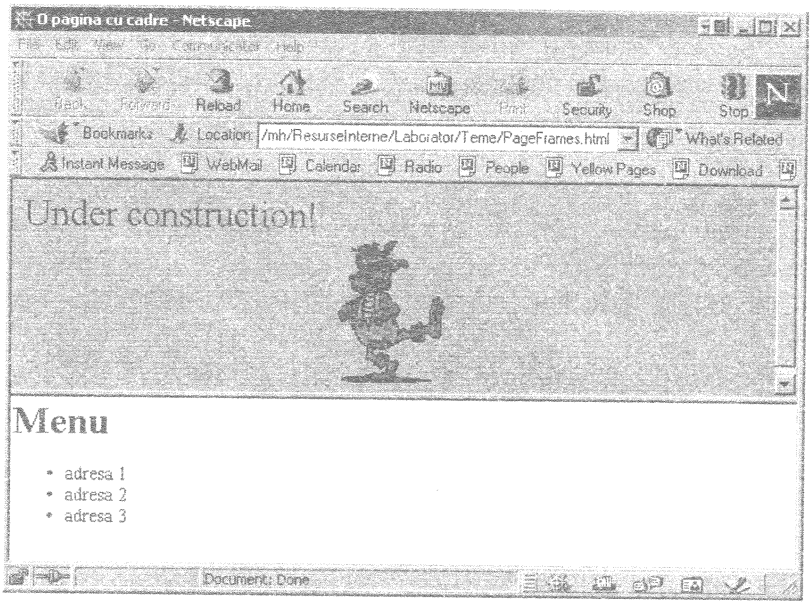


Figura 4-17. O pagină cu două cadre orizontale.

Pentru o mai bună înțelegere a utilizării cadrelor, să urmărim exemplul următor, salvat în fișierul `PageFrames.html`, în timp ce fișierul `Framestuff.html` reprezintă documentul cu anunțul Under construction discutat anterior iar fișierul `Menu.html` reprezintă o posibilă listă de legături etc.

```
<HTML>
<HEAD>
 <Title>O pagină cu cadre</Title>
</HEAD>
<FRAMESET rows="*,130">
 <FRAME name ="main" src ="Framestuff.html">
 <FRAME name="menu" src ="Menu.html"
        marginwidth =0
        marginheight =0
        scrolling ="no"
        noresize>
</FRAMESET>
<BODY>
<font size=7>
<a href="Framestuff.html">Resources</a>
```

```
</font>
</BODY>
</HTML>
```

Observație: elementele care, în mod normal, pot apare în secțiunea `BODY` *nu trebuie introduse în fața primului element* `FRAMESET`*, în caz contrar elementul* `FRAMESET` *fiind ignorat.*

Cadrele nu sunt întotdeauna o soluție agreată de utilizatori, deoarece:

- necesită încărcarea informației specifice mai multor documente, ceea ce impune un timp de afişare mai mare, în special clienților ce folosesc dial-up;
- conduc la reducerea spațiului de afişare a conținutului de detaliu;
- pot impune dificultăți de navigare prin aglomerarea spațiului vizualizat, în special în cazul utilizării prea multor cadre în aceeaşi fereastră.

Cadrele permit împărțirea ferestrei de navigare în mici ferestre, fiecare afişând o anumită pagină Web. Această metoda poate fi însă folositoare:

- în cazul navigării într-un site Web, prin păstrarea continuă a unei referințe constante (un index sau o tablă de materii) într-o secțiune a fereastrei pe ecran, în timp ce se parcurg diferite pagini ale site-ului în alt cadru;
- pentru păstrarea continuă a unor anunțuri (reclame, mesaje de ajutor etc.) pe ecran, în timpul navigării pe un site.

O caracteristică unică a cadrelor HTML este aceea că necesită un fişier HTML complet separat. Introducerea cadrelor în HTML a condus la două tipuri distincte de documente HTML:

- documente frameset (layout): sunt fişiere HTML care conțin doar informația asupra modului de împărțire a ferestrei de navigare. Nu conțin nici o informație despre conținutul fiecărei ferestre: text, desene, hyperlegături etc.
- documente body: sunt documentele HTML obişnuite, ce conțin toată informația despre pagina Web, folosind etichete HTML obişnuite. Fiecare cadru are propriul *body document,* care produce pagina Web vizibilă în cadrul respectiv. În acest sens, cadrele reprezintă o modalitate de a vizualiza mai multe pagini Web pe ecran în acelaşi timp.

Ambele tipuri de documente (frameset şi body) contin header-e care descriu titlul paginii Web. Diferența dintre ele este dată de

restul documnentelor, conținute în etichetele *body tags* sau *frameset tags*.

Să considerăm exemplul cu trei cadre. Codul ce produce o pagină împărțită

este descris de:

- *documentul frameset*, care se numeşte "a.html";
- *documentele body,* ce descriu conținutul fiecărui cadru, şi se numesc "x.html", "y.html" si "z.html".

```
Fişierul a.html:
<html>
<head>
<title> A </title>
</head>
<frameset rows = *,*>
 <frame src = "x.html" name = "upper">
 <frameset cols = *,*>
  <frame src = "y.html" name = "lower_left">
  <frame src = "z.html" name = "lower_right">
 </frameset>
</frameset>
</html>
```

Formatul documentului *frameset* este similar unui *body document*. Acesta începe şi se termină cu etichetele `<html>` `</html>` şi conține `header` şi `title`. În schimb, în timp ce în *body document*, codul ce descrie conținutul paginii Web este plasat între etichetele `<body>` şi `</body>`, în documentul `frameset`, informația este plasată între etichetele `<frameset>` şi `</frameset>` (ce pot fi imbricate: incluse unele în altele). Etichetele `<frameset>` declară împărțirea cadrelor, iar etichetele `<frame>` sunt folosite pentru definirea cadrelor (etichetele `<frame>` nu necesită perechea de închidere `</frame>`).

4.6.1. Etichete `<frameset>`

Etichetele de declarare `<frameset>` reprezintă primul pas în crearea cadrelor şi descriu divizarea ferestrei în cadre mai mici. Fiecare astfel de declarație poate împărți fereastra în rânduri **sau** în coloane, dar nu în ambele în acelaşi timp. Pentru obținerea unei ferestre

, trebuie intâi împărțită fereastra în două rânduri (top şi bottom), apoi rândul bottom (de jos) este împărțit în două coloane (lower_left şi lower_right). Dimensionarea fiecărui

cadru poate fi realizată prin specificarea numarului de pixeli, prin valoarea procentuală din dimensiunea ferestrei sau prin dimensiuni relative.

Spre exemplu:

```
<frameset rows = 30, 33%, *, 2*>
```

divide fereastra în patru cadre orizontale, fiecare având dimensiunile enumerate în listă: primul rând are 30 pixeli înălțime, al doilea are 33% din înălțimea totală a ferestrei. Folosirea wildcard-ului "*" se referă la "restul ferestrei": deci restul ferestrei este împartit în alte doua cadre, ultimul dintre ele fiind de două ori mai mare decât primul. Figura 4-18 arată rezultatul acestei declarații (dimensiunea de 30 pixeli va fi caracterizată de mărimea ferestrei de navigare):

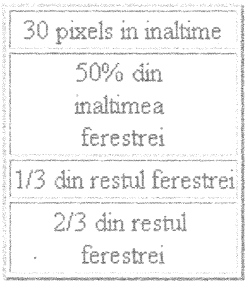


Figura 4-18. Un model de împărțire a unei ferestre în cadre orizontale.

Un rezultat asemănător poate fi obținut prin utilizarea parametrului `cols =` în loc de `rows =` rezultând o fereastră descrisă ca în figura 4-18, dar segmentată pe verticală. Dacă fereastra a fost partajată incorect (mai ales prin specificarea dimensiunii în pixeli), rezultatul va diferi de la browser la browser. Astfel, cel mai avantajos mod de lucru este prin procentaj şi cel putin un specificator "*". În fişierul `a.html` spre exemplu, fereastra este divizată în două rânduri egale prin `rows = *,*` (similar cu `"50%, 50%"` sau `"50%,*"`).

Observație: fiecare cadru poate fi la rândul său împărțit în coloane sau rânduri (dacă fereastra a fost inițial împărțită în coloane, fiecare cadru rezultat va fi împărțit apoi în rânduri. Există două modalități de segmentare a cadrelor: *încuibare directă* (direct nesting) şi *indirectă* (indirect nesting).

Încuibarea directă a cadrelor (direct nesting) presupune cadrele divizate în acelaşi document, rezultând cadre distincte. Fişierul `a.html` descris anterior este un caz de încuibare directă.

Încuibarea indirectă (indirect nesting) presupune cadrele inițial împărțite într-un prim document frameset, viitoarele divizări fiind definite în documente frameset separate. Are ca avantaj faptul că unele cadre pot fi grupate pentru a funcționa ca un singur frame. Codul sursă pentru cazul încuibării indirecte este:

```
filename: a.html
<frameset rows=50%,*>
 <frame src = "x.html"
        name = "upper">
 <frame src = "yz.html"
        name = "lower">
</frameset>
```

```
filename: yz.html
<frameset cols = 50%,*>
 <frame src = "y.html"
        name="lower_left">
 <frame src="z.html"
       name="lower_right">
</frameset>
```

Ambele metode produc același rezultat; diferența constă în faptul că al doilea caz (indirect nesting) permite tratarea celei de-a doua jumătăți a ecranului atât ca două cadre distincte (numite lower_left şi lower_right) cât şi ca un singur cadru (numit lower), putând fi actualizat mai mult de un singur frame la un moment dat! (vezi secțiunea targeting). Dezavantajul încuibării indirecte constă în faptul că trebuie să fie scrise mai multe documente frameset. Deci, încuibarea indirectă trebuie folosită doar dacă se dorește ca o hyperlegătură să actualizeze mai mult de un cadru în același timp.

4.6.2. Etichete `<frame>`

Odată ecranul împărțit în cadre, este necesară definirea fiecăruia: câte o singură etichetă `<frame>` pentru fiecare cadru declarat în `<frameset>`. Dacă nu sunt declarate, cadrele nedefinite vor fi goale. Cadrele se declară în ordinea:

- de sus în jos pentru rânduri;
- de la stânga la dreapta pentru coloane.

Rolul etichetei `<frame>` este acela de a lega respectivul cadru cu un *body document* declarat prin comanda `"src ="` (similar legării hyperlegăturilor sau imaginilor). O caracteristică suplimentară a etichetei `<frame>` este denumirea fiecărui cadru pentru o viitoare referire.

4.6.3. Ancorarea cadrelor

Fără cadre, folosirea unei hyperlegături implică schimbarea întregii ferestre prin aducerea conținutului paginii destinație a legăturii. Prin folosirea cadrelor, actualizarea se poate realiza pe cadre individuale, colecții de cadre sau pe întreaga fereastră, prin folosirea ancorelor (având drept destinație numele cadrelor specificate prin definiții `<frame name=". .">`). Ancorele sunt specificate în declarația hyperlegăturii din **body document**.

Spre exemplu, să presupunem ca body documentul `z.html` conține hyperlegătura:

```
<a href ="b.html" target="upper">Click here</a>
```

Când se selectează hyperlegătura (figura 4-19) în cadrul din dreapta (unde este localizat `z.html`), cadrul de sus se va schimba din `x.html` în `b.html`. Deci, prin alegerea unei hyperlegături într-un cadru, se poate actualiza un alt cadru fără modificarea celorlalte. Analog, destinația poate fi aleasă "lower_left" sau "lower_right" pentru actualizarea cadrelor din partea de jos a ferestrei. Pentru modificarea tuturor cadrelor de jos trebuie folosită încuibarea indirectă cu specificarea destinației "lower".

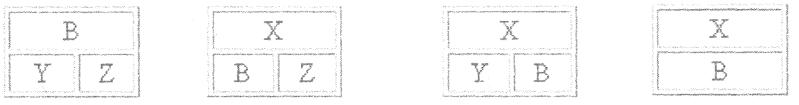


Figura 4-19. Destinații pentru `b.html`.

Unui frame i se poate da orice nume: în mod normal, se urmăreşte evidențierea poziției de pe ecran (ca mai sus) sau a funcției ("main", "contents" etc.). Există şi nume de ancore definite implicit, care nu trebuie folosite în eticheta `<frame>` (încep cu "_"):

- `"_self"` actualizează cadrul ce-l conține (este implicit dacă nu se specifică nici un target);
- `"_parent"` actualizează cadrul părinte dacă hyperlegătura este într-un cadru derivat (spre exemplu în cazul încuibării indirecte, "lower_left" şi "lower_right" sunt cadre derivate ale cadrului "lower");
- `"_top"` actualizează întreaga fereastră, indiferent de modul de partajare;
- `"_blank"` crează o fereastră nouă.

4.6.4. Caracteristici suplimentare ale cadrelor

În interiorul etichetei `<frameset ...>` pot fi utilizate combinații între parametrii:

```
rows = size1, size2, ..., size_n
```

Proprietatea împarte fereastra sau cadrul părinte în n *rânduri. Dimensiunea poate fi specificată în pixeli, procent din înălțimea ferestrei (%), sau dimensiuni relative (*).*

```
cols = size1, size2, ..., size_n
```

Proprietatea împarte fereastra sau cadrul părinte în n coloane. Dimensiunea poate fi specificată în pixeli, procent din lățimea ferestrei (%), sau dimensiuni relative (*).

`border = pixels`

Proprietatea caracterizează grosimea chenarului (în pixeli) ce defineşte cadrele. Pentru eliminarea completă a chenarului, `border=0` (creând o aparentă continuitate între cadre).

`bordercolor = "#rrggbb"`

Culoarea barei de margine în cod RGB.

În interiorul etichetei `<frame ...>` pot fi utilizate combinații între parametrii:

`src = "body_document"`

Specifică adresa URL pentru documentul body al fiecărui cadru.

`name = "frame_name"`

Proprietate ce permite ancorarea cadrelor individuale prin hyperlegături în oricare dintre cadrele definite anterior.

`noresize`

Invalidează posibilitatea ca utilizatorul să redimensioneze cadrul. Dacă nu este folosită, utilizatorul îşi poate redimensiona cadrul prin drag asupra marginii cadrului.

`scrolling = "yes/no/auto"`

În cazul în care conținutul unui cadru este mai mare decât cadrul, este necesară o bară de defilare. Opțiunea "yes" forțează prezența unei bare de defilare chiar dacă nu e necesar, "no" nu va permite deloc crearea barei de defilare, "auto" produce apariția barei de defilare doar dacă o impune raportul dimensiunilor cadrului şi conținutului documentului (subunitar). Valoarea implicită este "auto".

`marginheight = pixels`
`marginwidth = pixels`

Proprietăți care permit spațierea între marginea cadrului şi conținutul său (în pixeli).

4.6.5. Aplicații de navigare Web ce nu permit vizualizarea cadrelor

Dacă un browser ce nu detectează cadre deschide o fereastră ce conține declarații frames, rezultatul este o pagină albă (versiunile actuale ale browser-elor Netscape Navigator şi InternetExplorer nu fac parte din această categorie). În astfel de cazuri este preferabilă includerea etichetelor `<noframes>..</noframes>` (ignorate de celelalte browser-e). Unii proiectanți Web folosesc aceste etichete pentru apelarea unor copii ale site-ului ce nu utilizează cadre (deşi este o variantă dificil de întreținut), în timp ce în marea majoritate a cazurilor se inserează un mesaj prin care se anunță că se folosesc cadre sau se face o legătură la body documentul principal. Etichetele `<noframes>` sunt funcționale doar în cadrul creat de `<frameset>` şi nu pot fi folosite într-un body document.

Implementarea cea mai obişnuită a acestei etichete într-un frameset document este următoarea:

```
<frameset ...>
<frame ...>
...
...
<noframes>
This document requires a frames-capable browser.
We apologize for the inconvenience!
</noframes>
</frameset>
```

4.7. Forme HTML. Manipularea formelor HTML

Chiar dacă nu ați creat încă forme Web, acestea au putut deja fi folosite ca utilizatori Web (în cazul căutărilor pe Internet sunt prevăzute, spre exemplu, forme care permit personalizarea cheilor de căutare; multe site-uri Web cer informații de înregistrare; alte site-uri au implementate linii chat). Formele Web (sau formele HTML), reprezintă cel mai simplu mod de a transforma o pagină Web, dintr-o publicație on-line, într-un instrument interactiv.

4.7.1. Ce este o formă HTML?

O formă HTML este o secțiune a unui document Web în care utilizatorul poate introduce informații. Acestea sunt trimise către serverul Web, unde pot fi înregistrate într-o bază de date, pentru utilizări ulterioare, sau pentru controlul informațiilor returnate utilizatorului.

4.7.2. Pentru ce sunt indicate formele HTML?

Formele HTML pot realiza:

- introducerea de text sau alegerea dintr-o paletă de opțiuni;
- colectarea mai multor elemente de informație;
- restrângerea răspunsurilor utilizatorului la un set de valori bine definite;
- structurarea informației furnizate de utilizator pentru procesarea automată;
- implementarea GUIs (Graphical User Interfaces) pentru aplicații de rețea sau pentru navigarea pe site-uri Web;

4.7.3. Ce nu poate face o formă HTML?

Formele HTML nu pot furniza o interfață utilizator complet interactivă, ci doar construiesc o interogare sau un mesaj de răspuns. Nu există modalități de a controla prin forme informația introdusă în câmpurile text, ci doar de a furniza o modalitate de introducere a textului utilizatorului. Manipularea informației introduse de utilizator în forme necesită, în general, programe CGI special concepute pentru a o procesa.

4.7.4. Crearea unei forme HTML

Se realizează prin etichete HTML (prin adăugarea şi selectarea proprietăților unei forme, folosind un instrument de creare a paginilor HTML, de tipul Netscape PageComposer sau Microsoft FrontPage, sau chiar utilizând un simplu editor text).

Un exemplu de forma HTML

Într-o sursă HTML, o formă este semnalizată prin eticheta: `<FORM>...</FORM>`. Dacă avem un program ce va procesa datele trimise, de forma (`form_handler`, cel mai ades un program CGI), se va adăuga formei atributul `ACTION="url"` pentru a indica poziția programului şi atributul `METHOD=reqtype` pentru metoda de transmisie.

> *Observație: În toate exemplele care urmează, atributela opționale sunt semnalizate prin parantezele pătrate [ ] şi se folosesc pentru a suplimenta proprietățile unei forme.*
>
> *Aproape toate formele trebuie să fie descrise prin atributul* `NAME` *(identificator al componentei formei).*

Un exemplu de creare a unei forme simple este dat în cele ce urmează (figura 4-20):

```
<FORM ACTION="register.cgi" METHOD=POST>
```

```
<DL>
<DT>Your full name:
<DD><INPUT TYPE=TEXT NAME="fullname" SIZE=60
MAXLENGTH=180>
<DT>Your e-mail address:
<DD><INPUT TYPE=TEXT NAME="email" SIZE=60
MAXLENGTH=180>
</DL>
<P><INPUT TYPE=SUBMIT VALUE="Register">
</P></FORM>
```

Etichetele `<FORM> ... </FORM>` grupează câmpurile de introducere împreună şi defineşte *cum* şi *unde* vor fi procesate. Etichetele specifice unei liste de definiţie impun o mai bună structurare a modului de împărţire a specificaţiilor şi conţinutului dialogului cu utilizatorul. Etichetele `<P>..</P>` impun ca butonul `submit` să fie tratat ca buton separat.

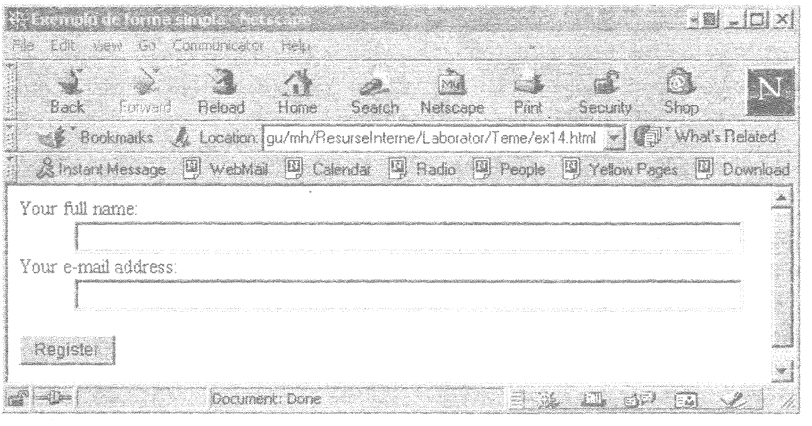


Figura 4-20. O simplă formă HTML.

Observaţie: Cele mai multe etichete HTML (ca `<P>`, `<PRE>`, `<BR>`, `<DL>`*) pot fi utilizate între* `<FORM>` *şi* `</FORM>` *pentru controlul aspectului formei.*

Folosirea etichetelor `<DT>` *şi* `<DD>` *în interiorul unor liste (marcate prin* `<DL> .. </DL>`*) conduce la o mai clară identificare a asocierilor text de descriere a formelor. În general, însă, se preferă etichete* `<P>..</P>` *pentru a grupa forme, şi etichete* `<BR>` *pentru a le separa.*

4.7.5. Componente ale unei forme HTML

În cadrul etichetelor `<FORM>..</FORM>` sunt valabile următoarele etichete HTML (HTML Form Tags: componente ale formei):

INPUT TYPE=TEXT

```
<INPUT [TYPE=TEXT] NAME="text-identificator"
[SIZE=nn] [MAXLENGTH=nn] [VALUE="default text"]>
```

O etichetă INPUT cu atributul TYPE=TEXT furnizează utilizatorului un prompt pentru o singură linie de text. Eticheta trebuie să aibă atributul NAME. Atributul SIZE se foloseşte pentru specificarea numărului de caractere care să caracterizeze dimensiunea ferestrei de introducere. Atributul MAXLENGTH se foloseşte pentru limitarea numărului de caractere de intrare. Dacă valoarea MAXLENGTH este mai mare decât valoarea SIZE, atunci browser-ul va permite derulare (scroll) pe textul de intrare. Atributul VALUE se foloseşte pentru afişarea unui text de intrare implicit, ce va fi afişat odată cu forma şi va putea fi modificat prin editare. În exemplul anterior (figura 4-20) este ilustrat un caz de utilizare INPUT TYPE=TEXT.

INPUT TYPE=SUBMIT

```
<INPUT TYPE=SUBMIT [NAME="button-id"]
[VALUE="Button label text"]>
```

Eticheta INPUT cu parametrul TYPE=SUBMIT furnizează un buton care transmite informaţia din forma completată către adresa URL dată ca atribut ACTION în eticheta <FORM>. Informaţia este transmisă prin tipul de apel HTTP specificat de atributul METHOD al formei.

O formă poate avea mai multe butoane SUBMIT, identificabile prin valoarea din atributul opţional NAME. Atributul NAME va fi transferat prin datele formei la apelul programului CGI pentru a permite mecanismului de procesare a datelor formei să determine ce buton a fost folosit de utilizator. În exemplul anterior (figura 4-20) este ilustrat un caz de utilizare a unui buton SUBMIT

INPUT TYPE=RESET

```
<INPUT TYPE=RESET [VALUE="Button label text"]>
```

O etichetă cu parametrul TYPE=RESET furnizează un buton care şterge forma şi setează conţinutul acesteia la valorile iniţiale.

> *Spre deosebire de cele mai multe etichete, în care atributul* NAME *este obligatoriu, în cazul* INPUT TYPE=SUBMIT acesta *este opţional, iar în cazul* INPUT TYPE=RESET *nu există.*

În continuare vom exemplifica (figura 4-21) un simplu buton de reset.

```
<FORM ACTION="choose.cgi" METHOD=POST>
<P>E-mail address: <INPUT TYPE=TEXT NAME="email"
SIZE=60 MAXSIZE=180> </P>
<P>Please add me to the mailing list. </P>
<P> I am: </P>
<OL>
<LI><INPUT TYPE=RADIO NAME="employer"
VALUE="private" chECKED>Employed in the private
sector
<LI><INPUT TYPE=RADIO NAME="employer"
VALUE="public">Employed in the public sector
<LI><INPUT TYPE=RADIO NAME="employer"
VALUE="self">Self-employed
<LI><INPUT TYPE=RADIO NAME="employer"
VALUE="unemployed">Unemployed
</OL>
<P><INPUT TYPE=SUBMIT VALUE="Continue">
<INPUT TYPE=RESET VALUE="Clear form">
</P>
</FORM>
```

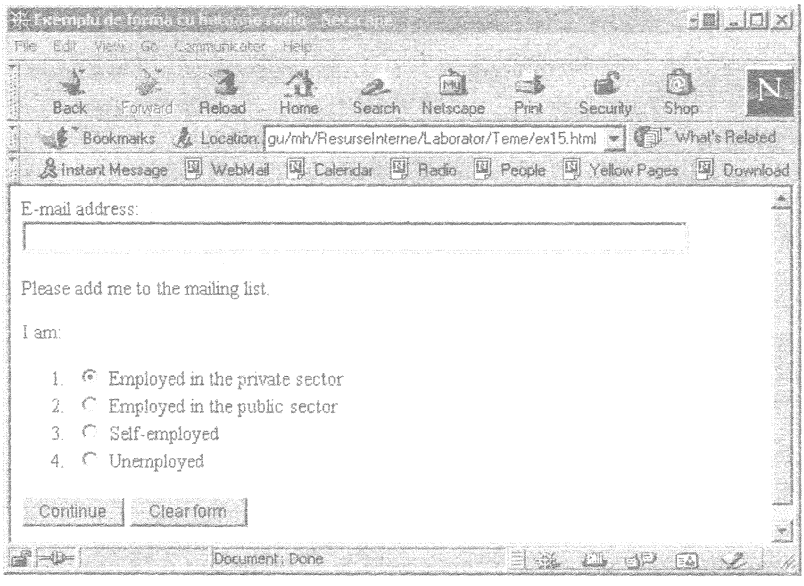


Figura 4-21. Exemplu de forma cu butoane radio.

INPUT TYPE=RADIO

```
<INPUT TYPE=RADIO NAME="radio-set-id"
VALUE="choice-id" [chECKED]>
```

Este o formă ce permite utilizatorului să aleagă dintr-un set de alternative. Fiecare buton radio din setul de alternative din formă are aceeaşi valoare a atributului. Evident, un singur buton radio poate fi selectat o singura dată (opţiune unică). Atributul VALUE specifică datele transmise. Butoanele radio trebuie să folosească acelaşi atribut NAME dar să aibă diferite atributele VALUE. Atributul CHECKED marchează un buton radio din listă, ca opţiune implicită. În exemplul anterior (figura 4-21) am evidenţiat folosirea butoanelor radio ca selecţie de tipul *o variantă din mai multe*.

INPUT TYPE=CHECKBOX

```
<INPUT TYPE=chECKBOX NAME="box-set-id"
VALUE="choice-id" [chECKED]>
```

O etichetă INPUT cu atributul TYPE=CHECKBOX oferă utilizatorului posibilitatea de a alterna între "on" şi "off". Este oarecum similar butoanelor radio, cu diferenţa că, oricâte componente checkboxes pot fi selectate sau nu. Un checkbox selectat ("on") la transmiterea formei îşi va transmite propria valoare VALUE pentru componenta respectivă. Mai multe componente checkboxes pot fi grupate (ca şi butoanele radio), prin utilizarea aceluiaşsi atribut NAME. Datele transmise în formăa sunt cele specifice componentelor "on" (selectate), separate prin virgulă. Pentru o listă de checkbox, trebuie să se includă pentru fiecare opţiune un text care să expliciteze semnificaţia alegerii.

În continuare vom exemplifica (figura 4-21) utilizarea componentelor checkboxes.

```
<FORM ACTION="feedback.cgi" METHOD=POST>
<P> Please tell us what you thought of this Web
site. Select the checkboxes which you agree with:
</P>
<UL> <LI><INPUT TYPE=chECKBOX NAME="opinion"
VALUE="understandable">
The text was understandable.
<LI><INPUT TYPE=chECKBOX NAME="opinion"
VALUE="navigable"> I found it easy to find my way
through the Web site.
<LI><INPUT TYPE=chECKBOX NAME="opinion"
VALUE="stylish">
I was impressed by the style and presentation.
</UL>
<P><INPUT TYPE="HIDDEN" NAME="pages"
VALUE="brochure">Please add any other comments:
<TEXTAREA NAME="feedback" ROWS=5 COLS=40>
I think your brochure is:
```

```
</TEXTAREA>
</P>
<P>
<INPUT TYPE=SUBMIT VALUE="Send comments">
<INPUT TYPE=RESET VALUE="Clear form">
</P>
</FORM>
```

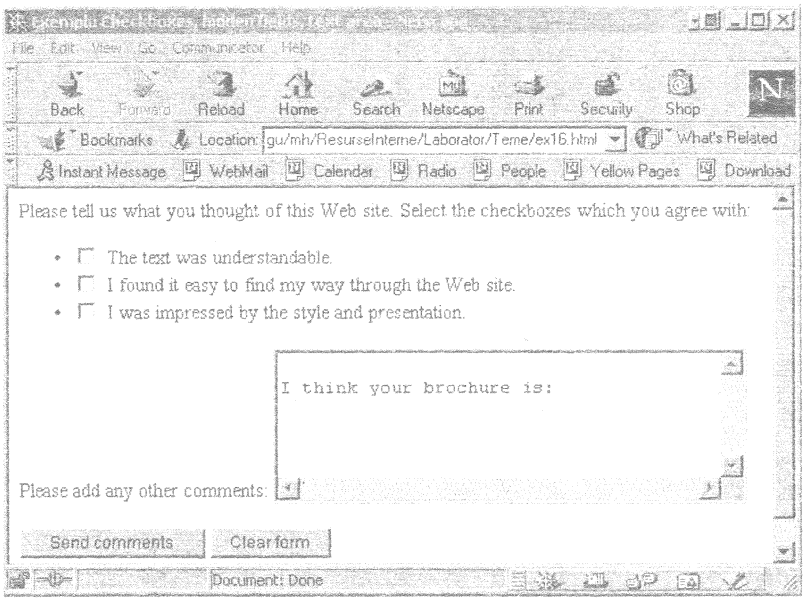


Figura 4-22. Un exemplu de utilizare a componentelor: checkboxes, hidden fields, text area.

INPUT TYPE=IMAGE

```
<INPUT TYPE=IMAGE NAME="image-id" SRC="image-url"
[ALIGN=alignment] >
```

Eticheta de forma `INPUT TYPE=IMAGE` este similară etichetei `IMG`. Folosirea ei are ca rezultat afişarea imaginii de la locaţia `SRC` cu opţiunea `ALIGN`. O astfel de formă are două caracteristici ce o fac utilă:

- la clic pe image, datele formei completate sunt transmise analog unei componente `INPUT TYPE=SUBMIT`;
- permite utilizatorului să aleagă o porţiune din imagine.

Coordonatele în pixeli ale punctului ales de utilizator, identificatorii *image-id.x* şi *image-id.y* (abscisa este transmisă ca fiind conţinutul atributului `NAME.x` iar ordonata ca `NAME.y`), sunt transmise împreună cu datele formei.

Exemplul următor ilustrează folosirea unei componente `INPUT TYPE=IMAGE` ca un buton `Submit` personalizat (figura 4-23) dar, în general, multe site-uri folosesc caracteristica referitoare la coordonatele în pixeli pentru a prezenta meniuri grafice, tabele de butoane sau hărți de navigare în site. Comportarea este similară etichetei referitoare la harta de imagini `<IMG ISMAP>`.

```
<FORM ACTION="select.cgi" METHOD=GET>
<H2>Choose which software to download:</H2>
<P><SELECT NAME="package" SIZE=3>
<OPTION VALUE="text" SELECTED>Text viewer
<OPTION VALUE="image">Image viewer
<OPTION VALUE="movie">Movie player
<OPTION VALUE="audio">Sound player
<OPTION VALUE="editor">Media editor
</SELECT>
<SELECT NAME="platform">
<OPTION>IBM pc compatible
<OPTION>Macintosh (68000)
<OPTION>Macintosh (Power pc)
</SELECT>
<SELECT NAME="options" MULTIPLE>
<OPTION>License
<OPTION>Media
<OPTION>Documentation
</SELECT>
</P>
<P>
<INPUT          TYPE="IMAGE"          NAME="coords"
SRC="download.gif">
</P>
</FORM>
```

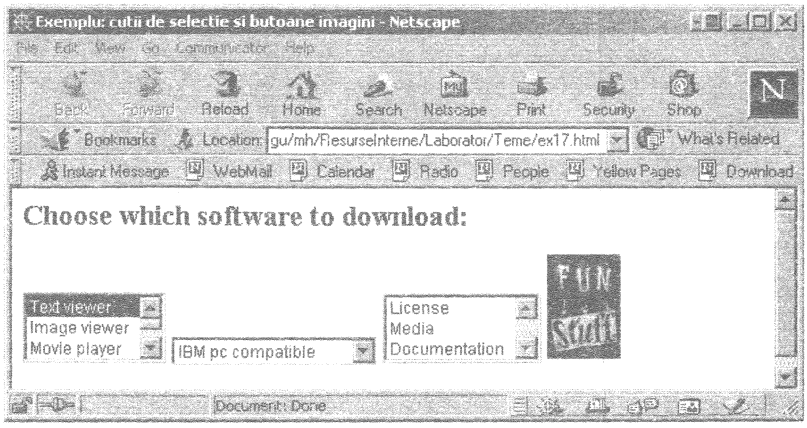


Figura 4-23. Cutii de selecție și butoane imagini.

INPUT TYPE=HIDDEN

`<INPUT TYPE=HIDDEN NAME="data-id" VALUE="hidden form data" >`

Eticheta `INPUT TYPE=HIDDEN` este o formă neuzuală, în sensul că nu apare în forma HTML afişată, fiind o convenție prin care proiectantul trimite atributul `VALUE` ca dată a formei, în special dacă programul de prelucrare este specificat în atributul `ACTION` al mai multor forme.

Un exemplu (figura 4-24) al utilizării componentei `INPUT TYPE=HIDDEN` este ilustrat în continuare:

```
<FORM ACTION="comments.cgi" METHOD=POST>
<P>In order that we may continue to provide a
high quality World Wide Web service, please take
the time to fill in this form.</P>
<DL>
<DT>Your surname (family name):
<DD><INPUT NAME="surname" SIZE=20 MAXLENGTH=60>
<DT>Your first name (given name):
<DD><INPUT NAME="forename" SIZE=20 MAXLENGTH=60>
</DL>
<P>Your title:
[<INPUT TYPE=RADIO NAME="title" VALUE="Mr"> Mr.]
[<INPUT TYPE=RADIO NAME="title" VALUE="Ms"> Ms.]
[<INPUT TYPE=RADIO NAME="title" VALUE="Mrs">
Mrs.]
[<INPUT TYPE=RADIO NAME="title" VALUE="Miss">
Miss.]
[<INPUT TYPE=RADIO NAME="title" VALUE="Dr"> Dr.]
[<INPUT TYPE=RADIO NAME="title" VALUE=""> Other.]
</P>
<P>Please tell us what you thought of this Web
site. Select the checkboxes which you agree with:
</P>
<UL>
<LI><INPUT TYPE=chECKBOX NAME="opinion"
VALUE="understandable">
The text was understandable.
<LI><INPUT TYPE=chECKBOX NAME="opinion"
VALUE="navigable">
I found it easy to find my way through the Web
site.
<LI><INPUT TYPE=chECKBOX NAME="opinion"
VALUE="stylish">
I was impressed by the style and presentation.
```

```
</UL>
<P>
<INPUT TYPE="HIDDEN" NAME="pages"
VALUE="brochure">
Please add any other comments:
<TEXTAREA NAME="feedback" ROWS=5 COLS=40>
I think your brochure is:
</TEXTAREA>
</P>
<P>
<INPUT TYPE=SUBMIT VALUE="Send comments">
<INPUT TYPE=RESET VALUE="Clear form">
</P>
</FORM>
```

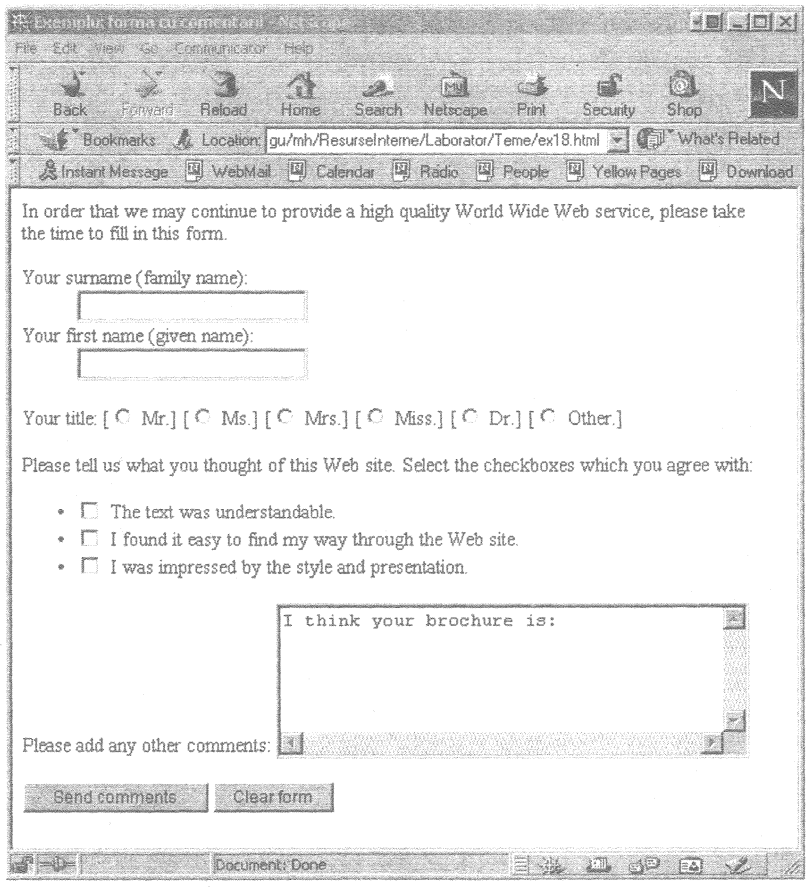


Figura 4-24. O formă pentru comentarii.

TEXTAREA

```
<TEXTAREA NAME="text-id" [COLS=nn] [ROWS=nn]>
default text</TEXTAREA>
```

Eticheta TEXTAREA prezintă o fereastră pentru editare de text pe mai multe linii (este un container HTML). Textul conținut între etichetele <TEXTAREA> şi </TEXTAREA> apare ca fiind un conținut implicit cazul prezentat în exemplul aferent figurii 4-22, pentru care conținutul implicit nu a fost încă modificat de utilizator).

Observație: un container HTML este o pereche de etichete, deschisă <AICI> şi închisă </AICI>, a cărei acțiune se aplică textului cuprins între etichete.

SELECT

```
<SELECT NAME="select-id" [SIZE=nn] [MULTIPLE]>
<OPTION [VALUE="choice-id"] [SELECTED]>1st choice
<OPTION>2nd choice
<OPTION>...
</SELECT>
```

Eticheta SELECT este o alternativă a butoanelor radio şi checkboxes, prezentând o listă de opțiuni într-o fereastră cu defilare (scrolling window). Dacă este folosit atributul MULTIPLE, componenta SELECT este comparabilă cu checkboxes, în sensul posibilității selectării mai multor variante. Atributul SIZE specifică numărul variantelor vizibile (controlând deci dimensiunea ferestrei de afişare). Atributul VALUE al fiecărei opțiuni OPTION selectate este transmis prin datele formei pentru procesare (dacă atributul este omis, se transmite conținutul opțiunii selectate). Dacă se selectează mai multe opțiuni, atributele VALUE sunt transmise împreună într-o listă, separate prin virgulă. În exemplul aferent figurii 4-23 este tratat un caz al utilizării etichetelor SELECT.

4.7.6. Concluzii

Formele HTML furnizează o modalitate simplă de a permite unui utilizator de browser web să transmită informații către site-ul Web vizitat, să caute eficient informații şi să interacționeze în Internet. Toate acestea, cu observația că este absolut necesar, din partea proiectantului Web, să caute, să aleagă şi să implementeze şi mijloace de securitate adecvate dimensiunii şi funcțiunii proiectului).

5. APLICAȚII JAVASCRIPT PENTRU DINAMIZAREA PAGINILOR HTML

5.1. Ce este JavaScript?

JavaScript este un limbaj de **script**-uri (liste de comenzi executate fără a interacționa cu utilizatorul) dezvoltat de Netscape. Este dedicat creării cu relativă ușurință a paginilor Web interactive. Vom studia în cele ce urmează câteva detalii despre *ce* și mai ales *cum* poate fi realizat cu JavaScript.

5.2. Executarea codului JavaScript

Pentru a executa coduri JavaScript avem nevoie de un browser cu capabilități JavaScript (spre exemplu orice Netscape Navigator – începând cu versiunea 2.0, sau Microsoft Internet Explorer – începând cu versiunea 3.0). Prin urmare, majoritatea utilizatorilor Internet au astăzi posibilitatea tehnică de a rula astfel de scripturi, ceea ce reprezintă un important motiv pentru a folosi JavaScript pentru dinamizarea paginilor HTML create.

Codul JavaScript se introduce direct în pagina HTML. Să studiem rezultatul unui prim exemplu de script în următoarea secvență:

```
<html>
<body>
<br>
This is a normal HTML document.
<br>
  <script language="JavaScript">
    document.write("This is JavaScript!")
  </script>
<br>
Back in HTML again.
</body>
</html>
```

La o primă vedere, această secvență pare a fi extrem de cunoscută. Noutatea apărută totuși este în rândurile:

```
  <script language="JavaScript">
    document.write("This is JavaScript!")
  </script>
```

Acestea sunt de fapt linii realizate în codul JavaScript. Pentru a observa rezultatul execuției acestui cod, trebuie salvată întreaga secvență ca un fișier HTML normal, urmând să fie încărcat într-un browser compatibil JavaScript.

În figura 5-1 se poate urmări ieşirea generată de acest exemplu: trei linii de mesaj text:

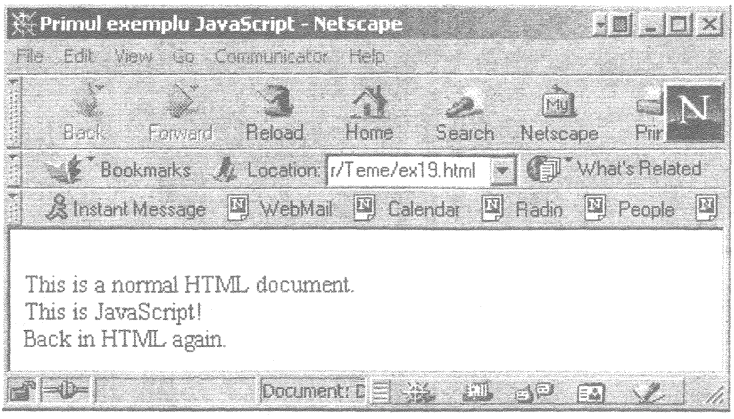


Figura 5-1. Primul exemplu JavaScript.

Bineînțeles, toate mesajele vizualizate în figura 5-1 pot fi realizate în cod HTML. Scopul exemplului este să scoată în evidență eticheta `<script>`. Secvențele cuprinse între etichetele `<script language="JavaScript">` şi `<script>`, oriunde în documentele HTML, vor fi interpretate de navigator ca un cod JavaScript. Una din cele mai importante şi utilizate funcții JavaScript este apelată în această secțiune: *document.write():* este în general apelată pentru scrierea în documentul current. Prin urmare, exemplu a evidențiat codul JavaScript care scrie textul *This is JavaScript!* într-un document HTML.

> *Observație: codul JavaScript nu poate fi ascuns de utilizatorul paginii! Orice navigator poate pune la dispoziție codul Javascript din cadrul documentului HTML, prin opțiunea* "View document source".

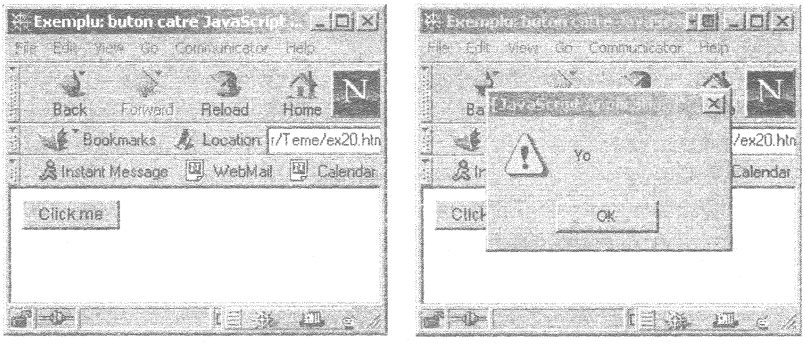
5.3. Evenimente

Tratarea evenimentelor reprezintă unul dintre obiectivele programării JavaScript, deoarece, în marea majoritate, acestea sunt determinate de acțiuni ale utilizatorilor. Spre exemplu, dacă utilizatorul apasă un buton (de mouse sau tastatură), se face, practic, apel la un eveniment *Click*. Dacă săgeata de mouse a fost deplasată deasupra unei hyperlegături, este apelat un eveniment *MouseOver*.

În scopul dinamizării paginilor HTML la acțiunile utilizatorului, , trebuie folosite tratările de evenimente JavaScript, deoarece codul JavaScript reacționează la evenimente.

Să considerăm următorul exemplu, care tratează evenimentul *onClick*:

```
<form>
<input type="button" value="Click me"
onClick="alert('Yo')">
</form>
```



(a) (b)

Figura 5-2. Apelul unei funcții alert JavaScript (b) prin tratarea unui eveniment relativ la un buton HTML (a).

În exemplul de mai sus, a fost declarat ca element de formă HTML un buton. Noutatea constă în apelul funcției `alert()` în cadrul etichetei `<input>`, la detectarea evenimentului `onClick`. Acest eveniment definește acțiunea care va fi executată dacă butonul este apăsat. Funcția alert permite deschiderea unei ferestre de atenționare (figura 5-2b), având ca mesaj afișat, parametrul de apel.

> *Observație: în acest caz, apelul codului JavaScript nu a fost realizat în interiorul etichetelor* `<script>`*, cum se procedează în mod uzual.*

O atenție specială trebuie acordată ghilimelelor folosite pentru semnalizarea parametrului de tip text. În apelul funcției alert apar ghilimelele simple: *'Yo'*. În schimb, la apelul funcției `document.write()`, au fost folosite ghilimelele duble. De fapt, pot fi folosite, pentru specificarea parametrilor de tip text, oricare din cele două tipuri de ghilimele. În interiorul funcției `alert` apar ghilimelele simple, deoarece aceasta este deja parametru apelului `onClick="alert('Yo')"`, motiv pentru care au fost folosite de fapt, imbricat, ambele tipuri.

Un apel care ar folosi doar un singur tip, de forma: `onClick="alert("Yo")"`, ar conduce la confuzii de interpretare, în sensul identificării parametrilor: spre exemplu, unde începe şi unde se termină parametrul de apel al evenimentului `onClick`. Acesta este motivul pentru care cele două tipuri de ghilimele trebuie să alterneze, indiferent de ordinea apariției. Prin urmare, apelul: `onClick='alert("Yo")'` este la fel de corect!

5.4. Funcții JavaScript

Marea majoritate a codurilor JavaScript inserate în documentele HTML se bazează pe funcții. Ideea este general valabilă în cazul funcțiilor: motivația utilizării funcțiilor este aceea de a grupa mai multe comenzi, într-o secvență care să fie folosită repetat, în mai multe secțiuni ale documentului.

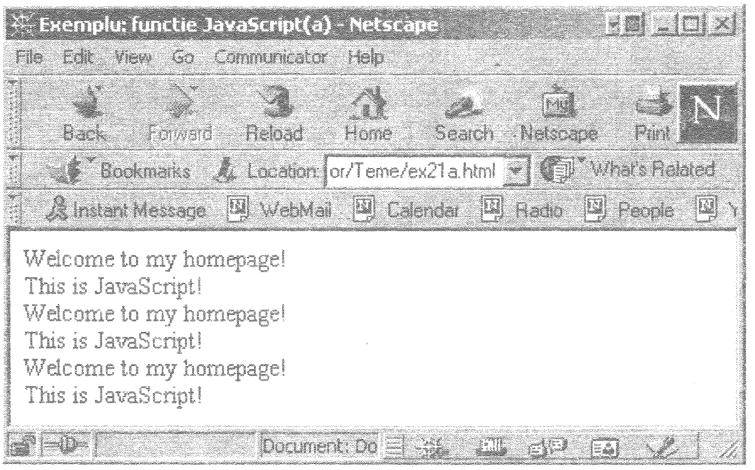


Figura 5-3. Apelul repetat al unui segment de scriere JavaScript.

Să considerăm exemplul următor, care afişează un anumit text de trei ori (figura 5-3):

```
<html>
<script language="JavaScript">
<!-- hide

document.write("Welcome to my homepage!<br>");
document.write("This is JavaScript!<br>");

document.write("Welcome to my homepage!<br>");
document.write("This is JavaScript!<br>");
```

```
document.write("Welcome to my homepage!<br>");
document.write("This is JavaScript!<br>");

// -->
</script>
</html>
```

Dar este eficientă scrierea unui cod similar sau chiar identic pentru a obține un rezultat? Evident, utilizarea apelului repetat al unei funcții care să încadreze secvența repetată de cod este mult mai eficientă, ca şi mai uşor de urmărit.

Codul următor va avea acelaşi rezultat, pe baza definirii şi folosirii funcției JavaScript `myFunction`:

```
<html>
<script language="JavaScript">
<!-- hide

function myFunction() {
  document.write("Welcome to my homepage!<br>");
  document.write("This is JavaScript!<br>");
}

myFunction();
myFunction();
myFunction();

// -->
</script>
</html>
```

Scriptul anterior are ca scop definirea funcției:

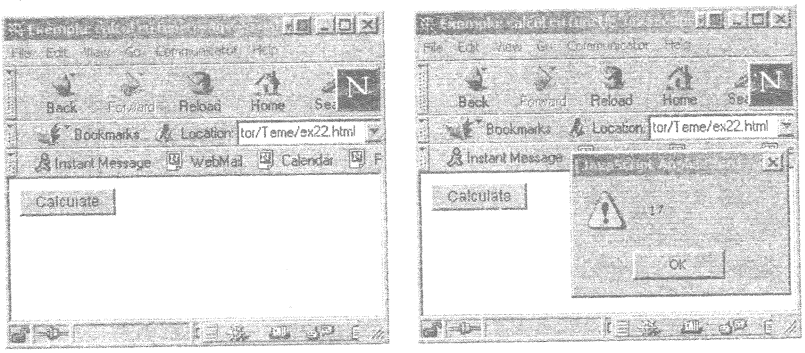
```
function myFunction() {
  document.write("Welcome to my homepage!<br>");
  document.write("This is JavaScript!<br>");
}
```

Cele două apeluri esențiale document.write() au fost încadrate în noua funcție, urmând să fie executate împreună printr-un singur apel al acesteia. Evident, rezultatul este identic, în cele două cazuri. În exemplul anterior erau necesare trei apeluri, deci vom scrie *myFunction()* de trei ori, în segmental corespunzător codului JavaScript, aceasta având ca rezultat execuția conținutului respectivei funcții de atâtea ori de câte ori a fost apelată.

Evident, exemplul de mai sus este extrem de simplu, utilizarea funcțiilor JavaScript având şi rolul, extrem de important, de generalizare prin folosirea parametrilor şi realizarea unui cod flexibil, dependent de contextual de apel.

Vom evidenția în exemplul următor folosirea funcțiilor JavaScript în combinație cu diferite evenimente (figura 5-4):

```
<html>
<head>

<script language="JavaScript">
<!-- hide

function calculation() {
  var x= 12;
  var y= 5;

  var result= x + y;

  alert(result);
}

// -->
</script>

</head>
<body>

<form>
<input type="button" value="Calculate"
  onClick="calculation()">
</form>

</body>
</html>
```



(a) (b)

Figura 5-4. Apelul unei funcții JavaScript de calcul.

Apăsarea butonului definit în forma anonimă (figura 5-4a) apelează funcția JavaScript `calculation()`. Pentru realizarea calculului matematic în corpul funcției, au fost folosite variabilele *x*, *y* și *result*. O variabilă se definește cu cuvântul cheie *var*.

Variabilele JavaScript sunt folosite pentru memorarea unor valori de tipuri diverse: numere, șiruri de caractere etc.

Linia `var result=x+y;` anunță browser-ul să creeze variabila *result* care să memoreze rezultatul *x+y* (spre exemplu, 5+12). După această operație, conținutul variabilei *result* este *17*. Comanda `alert(result)` este în acest caz (figura 5-4b) identică apelului `alert(17)`.

5.5. Ierarhizarea JavaScript a elementelor unui document HTML

Toate elementele unei pagini Web sunt organizate de JavaScript conform unei ierarhii, în care fiecare element este considerat un obiect. Fiecare astfel de obiect poate avea anumite proprietăți și metode. JavaScript propune astfel mijloacele prin care proprietățile unor asftfel de obiecte să fie accesate. Important este să fie înțeles modul prin care acestea să fie accesate, care depinde de ierarhia obiectelor HTML.

Să considerăm următorul document HTML (figura 5-5):

```
<html>
<head>
<title>My homepage</title>
</head>
<body bgcolor=#ffffff>

<center>
<img src="home.gif" name="pic1" width=200
height=100>
</center>

<p>

<form name="myForm">
Name:
<input type="text" name="name" value=""><br>
e-Mail:
<input type="text" name="email" value=""><br><br>
<input type="button" value="Push me"
name="myButton" onClick="alert('Yo')">
</form>
```

```
<p>
<center>
<img    src="ruler.gif"    name="pic4"    width=300
height=15>
<p>

<a href="http://www.stefan.org">My homepage</a>
</center>

</body>
</html>
```

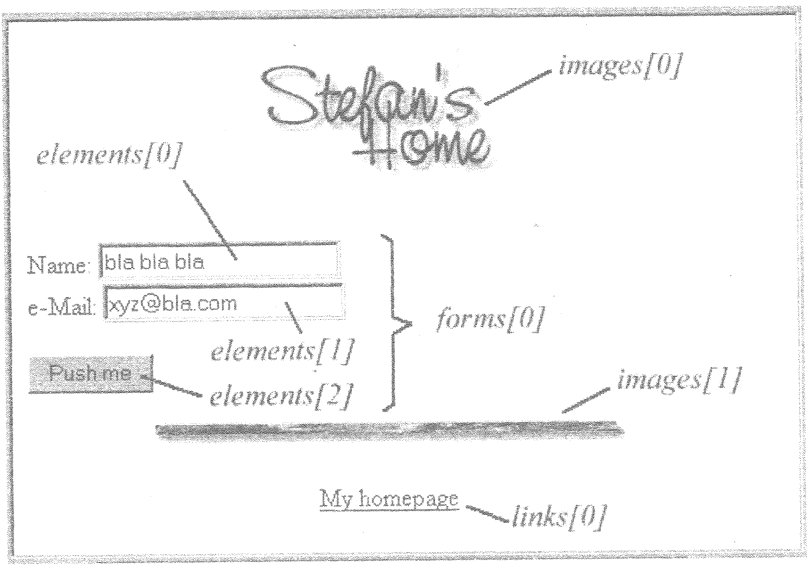


Figura 5-5. Un exemplu de document HTML pentru ierarhia JavaScript.

Documentul conține două imagini, o hyperlegătură şi o formă cu două câmpuri text şi un buton. Din punctul de vedere al ierarhizării JavaScript, fereastra browser-ului este un obiect *window*. Acest obiect conține anumite elemente (de exemplu, bara de stare). Într-o astfel de fereastră, poate fi încărcată o pagină HTML. O astfel de pagină este considerată un obiect *document*. Cu alte cuvinte, obiectul document reprezintă documentul HTML curent, încărcat de browser. Un obiect document este caracterizat de proprietăți specifice, spre exemplu culoarea de fundal a paginii. Toate obiectele HTML (hyperlegături, forme, imagini inserate) sunt considerate proprietăți ale obiectului document. Ierarhia JavaScript creată pentru documentul descris anterior este evidențiată în figura 5-6.

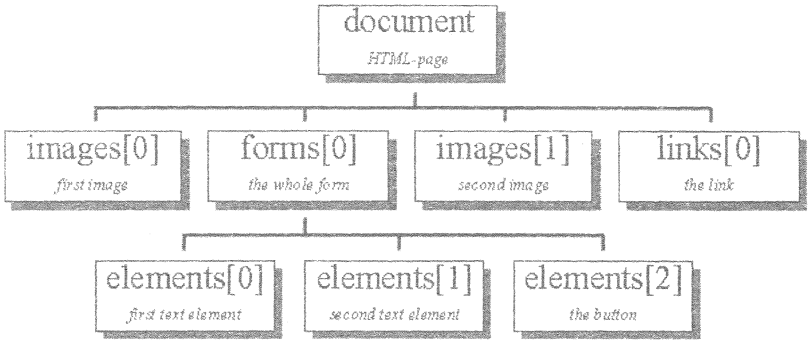


Figura 5-6. Ierarhia JavaScript a unui document HTML.

Din ierarhia prezentată în figura 5-6 putem identifica numele implicit al obiectelor documentului HTML. Spre exemplu, adresarea imaginii din document se obține astfel: plecând din vârful ierarhiei, primul document este *document.* Pentru a ajunge la prima imagine a documentului, trebuie să accesăm vectorul *images*, deci *images[0].* Prin urmare, accesul prin cod JavaScript a imaginii din document se realizează cu *document.images[0].*

Pentru a putea folosi ce a introdus utilizatorul în primul element al formei HTML, va trebui, similar, să identificăm calea de accesare a acestui element, din ierarhie. Pornind sin vârful ierarhiei, vom urma calea către primul element prin vectorul elements, prin urmare: *document.forms[0].elements[0*]. Rămâne să identificăm acea proprietate a unui element text al unei forme HTML care conține valoarea lui: *value*. În concluzie, linia de cod care preia valoarea introdusă de utilizator în elementul cu semnificația Name este:

```
name= document.forms[0].elements[0].value;
```

Şirul respectiv este memorat în variabila JavaScript *name*. Putem să folosim această variabilă în continuare, spre exemplu pentru a returna o formulă de salut: `alert("Hi " + name).`

Cu toată simplitatea şi claritatea ierarhiei Javascript, ar putea fi destul de greoi să adresăm prin indici numerici componentele unui document HTML cu multe elemente, spre exemplu: *document.forms[3].elements[17].* O astfel de problemă se rezolvă prin atributul unic NAME al fiecărui obiect HTML, Spre exemplu, în documentul considerat:

```
<form name="myForm">
Name:
<input type="text" name="name" value=""><br>
...
```

Prin urmare, *forms[0]* poate fi accesată şi prin *myForm*. Putem astfel să înlocuim:

```
name= document.forms[0].elements[0].value;
```

cu:

```
name= document.myForm.name.value;
```

Modalitatea de accesare devine astfel mai uşoară şi personalizată, mai ales în cazul paginilor cu multe obiecte, respectând însă cu strictețe denumirea atribuită obiectelor (nu se poate, spre exemplu, folosi *myform* în loc de *myForm*). Proprietățile obiectelor HTML, accesate prin ierarhia JavaScript pot fi folosite atât în operații de atribuire, citire sau calcul. Spre exemplu, în momentul creării unui document, se poate atribui o valoare inițială implicită unui element text al formei:

```
<form name="myForm">
<input type="text" name="input" value="insert
your text">
<input type="button" value="Write"
  onClick="document.myForm.input.value= 'Yo!'; ">
```

În codul descris în continuare vom identifica şi vom folosi o mare parte a proprietăților unor obiecte HTML. Rezultatul este descris în figura 5-7.

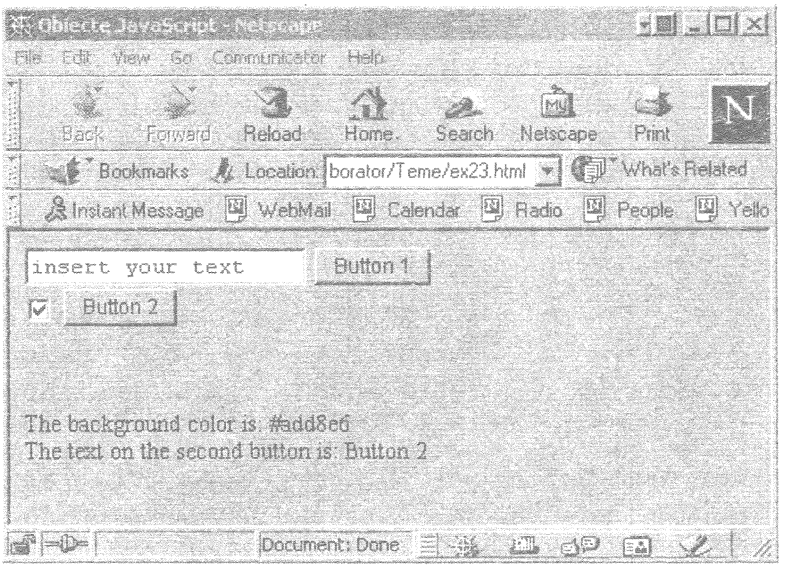
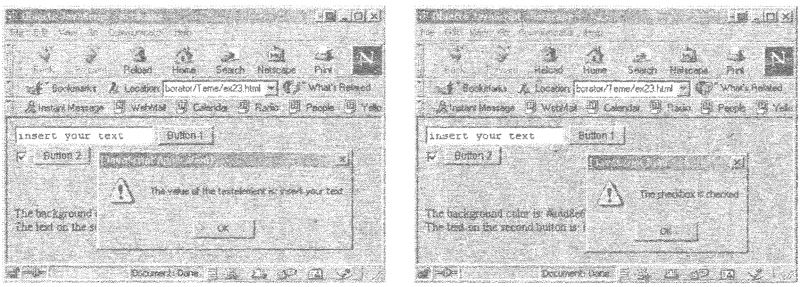


Figura 5-7. Un exemplu de control al proprietăților obiectelor unui document HTML prin funcții JavaScript.

```
<HTML>
<HEAD>
   <TITLE>Obiecte JavaScript</TITLE>

<script language="JavaScript">
<!-- hide

function first() {
  // deschide o fereastră de validare
  // conţinând textul din obiectul text
  alert("The value of the textelement is: " +
    document.myForm.myText.value);
}

function second() {
  // verifică starea obiectului checkbox

  var myString= "The checkbox is ";
  // este checkbox validat?
  if (document.myForm.myCheckbox.checked)
myString+= "checked"
  else myString+= "not checked";
  alert(myString);
}

// -->
</script>
</HEAD>

<body bgcolor=lightblue>

<form name="myForm">
<input  type="text"  name="myText"  value="insert
your text">
<input type="button" name="button1" value="Button
1"
  onClick="first()">
<br>
<input type="checkbox" name="myCheckbox" CHECKED>
<input type="button" name="button2" value="Button
2"
  onClick="second()">
</form>

<p><br><br>

<script language="JavaScript">
```

```
<!-- hide
document.write("The background color is: ");
document.write(document.bgColor + "<br>");
document.write("The text on the second button is:
");
document.write(document.myForm.button2.value);

// -->
</script>
</body>
</HTML>
```

În figura 5-8 putem identifica urmările acțiunii asupra celor două butoane, în urma cărora am putut procesa atributele elementelor HTML prin intermediul funcțiilor JavaScript ce au folosit ierarhia descrisă anterior pentru a le accesa.



(a) (b)

Figura 5-8. Rezultatele verificării prin funcții JavaScript a proprietăților obiectelor HTML: (a) la acționarea butonului 1; (b) la acționarea butonului 2.

5.5.1. Obiectul `location`

În afara obiectelor fereastră (`window`) şi `document`, un alt obiect extrem de important din punct de vedere al proprietăților este obiectul `location`. Acesta reprezintă adresa documentului HTML curent. Spre exemplu, dacă documentul încărcat este `http://www.xyz.com/page.html`, atunci `location.href` este chiar această adresă. Un aspect practic extrem de important este că, dacă vom atribui o nouă valoare către `location.href`, rezultatul va fi încărcarea noii adrese. În exemplul următor, apăsarea butonului va conduce la încărcarea noii adrese în fereastra curentă:

```
<form>
<input type=button value="Yahoo"
onClick="location.href='http://www.yahoo.com';">
</form>
```

O aplicație interesantă este controlul și personalizarea prin funcții JavaScript a proprietăților unei ferestre deschise ca urmare a tratării unui eveniment dependent de utilizator. Un exemplu este descris în codul următor și poate fi vizualizat în figura 5-9.

```
<HTML>
<HEAD>

<SCRIPT LANGUAGE="JavaScript">

<!-- Begin
function customize(form) {
var address = document.form1.url.value;
var op_tool=(document.form1.tool.checked== true) ? 1 : 0;
var op_loc_box=(document.form1.loc_box.checked==true)?1:0;
var op_dir = (document.form1.dir.checked == true)?1:0;
var op_stat = (document.form1.stat.checked == true)?1:0;
var op_menu  = (document.form1.menu.checked == true)?1:0;
var op_scroll = (document.form1.scroll.checked==true)?1:0;
var op_resize =(document.form1.resize.checked == true)?1:0;
var op_wid  = document.form1.wid.value;
var op_heigh = document.form1.heigh.value;
var option = "toolbar="+ op_tool +",location="+ op_loc_box
+",directories="
+  op_dir  +",status="+  op_stat  +",menubar="+  op_menu
+",scrollbars="
+ op_scroll +",resizable="  + op_resize +",width=" + op_wid
+",height="+ op_heigh;
var win3 = window.open("", "what_I_want", option);
var win4 = window.open(address, "what_I_want");
}
function clear(form) {
document.form1.wid.value="";
document.form1.heigh.value="";
}
// End -->
</SCRIPT>
</HEAD>

<BODY>

<CENTER>
<h4>Please  choose  from  the  following  selections  to
customize your window</h4>
<br>
<TABLE cellpadding=5 border><TR><TD><PRE>
<FORM name=form1  ACTION="javascript:" METHOD="POST">
<INPUT TYPE="text" NAME="url"
value="http://www.geocities.com" >: URL
<INPUT TYPE="checkbox" NAME="tool">: Toolbar
<INPUT TYPE="checkbox" NAME="loc_box">: Location
<INPUT TYPE="checkbox" NAME="dir">: Directories
<INPUT TYPE="checkbox" NAME="stat">: Status
<INPUT TYPE="checkbox" NAME="menu">: Menubar
<INPUT TYPE="checkbox" NAME="scroll">: Scrollbars
<INPUT TYPE="checkbox" NAME="resize">: Resizable
```

```
<INPUT TYPE="text"   NAME="wid" value= >: Width
<INPUT TYPE="text"  NAME="heigh" value=>: Height
<BR><CENTER>
<INPUT TYPE="button" VALUE="=ENTER="
 OnClick="customize(this.form)">
<INPUT TYPE="reset" VALUE="=RESET="
 onClick="clear(this.form)">
</PRE></TD></TR></TABLE>
</FORM>
</CENTER>
</BODY>
</HTML>
```

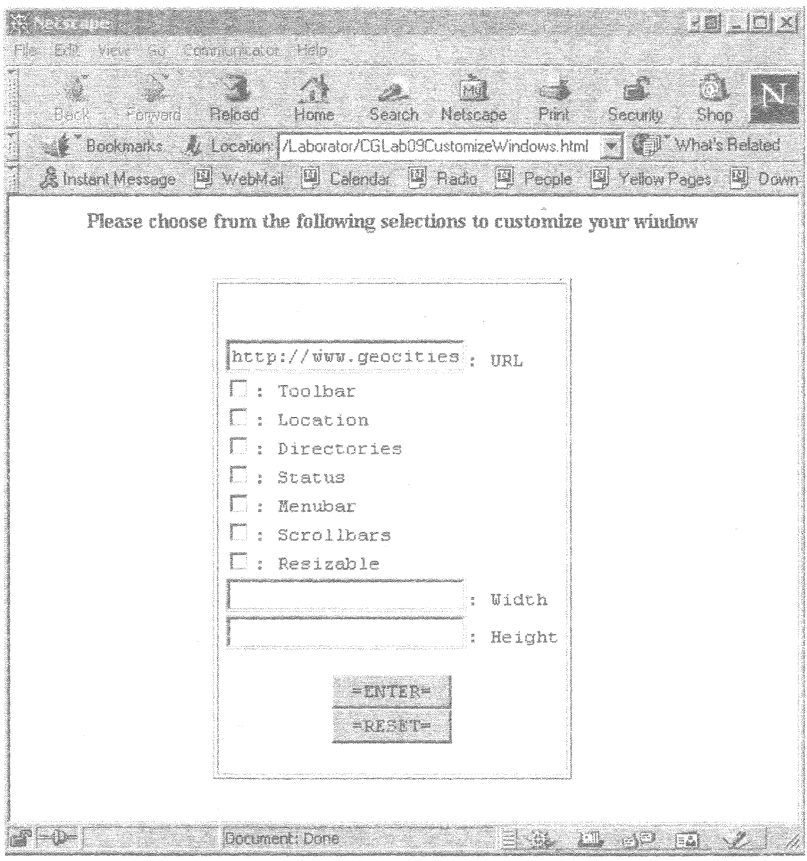


Figura 5-9. Deschiderea unei ferestre de navigare având caracteristicile setate de utilizator.

5.6. Ierarhizarea JavaScript a cadrelor HTML

Similar ideii de ierarhizare a elementelor unei pagini Web, cadrele sunt interpretate pe baza aceluiaşi principiu. Fie o fereastră HTML împărțită de două cadre:

```
<html>
<frameset rows="50%,50%">
  <frame src="page1.htm" name="frame1">
  <frame src="page2.htm" name="frame2">
</frameset>
</html>
```

Ierarhia JavaScript pentru cele două noi obiecte este următoarea:

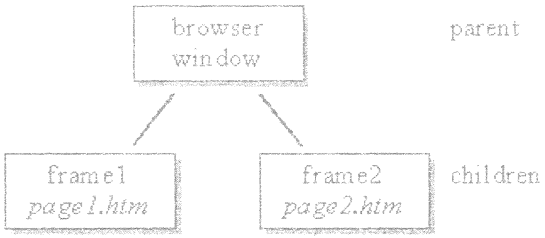


Figura 5-10. Ierarhia JavaScript pentru cadre HTML.

La baza ierarhiei se află fereastra browser-ului, obiectul părinte `window`. Cele două cadre reprezintă descendenții acestuia în ierarhie, cărora le sunt atribuite identificatorii unici `frame1` şi `frame2`. Prin intermediul acestora, informația poate fi accesibilă între diferitele elemente ale ierarhiei.

Una dintre problemele des întâlnite în gestionarea informației din cadre este deschiderea, într-unul din cadre, a unei pagini a cărei adresă se regăseşte ca hyperlegătură într-un alt cadru al ierarhiei. Este practic cazul meniurilor verticale sau hărților de site aflate într-un cadru, în timp ce un alt cadru este folosit pentru vizualizarea documentului cerut.

Practic, apar următoarele cazuri:

- Cerere de accesare într-un obiect părinte a informației unui cadru descendent;
- Cerere de accesare într-un obiect descendent a informației dintr-un obiect părinte;
- Cerere de accesare într-un obiect descendent a informației dintr-un alt obiect descendent.

Din punctul de vedere al ferestrei (identificată cu obiectul părinte), cele două cadre descendente se numesc `frame1` şi `frame2`, iar legătura către acestea este directă. Cu alte cuvinte, un script din

fereastra părinte (pagina care defineşte cadrele) poate accesa orice informaţie din cadrele descendente folosind direct numele acestora (figura 5-11). Spre exemplu, putem scrie:

```
frame2.document.write("A message from the parent
window.");
```

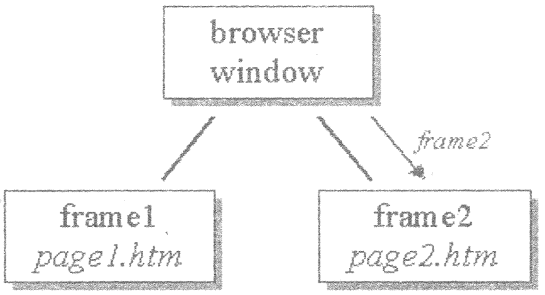


Figura 5-11. Accesarea informaţiei din cadrele descendente.

Sunt cazuri în care se doreşte accesarea, în sens invers, a informaţiei din documentul părinte într-un cadru descendent, spre exemplu dacă se doreşte renunţarea la divizarea în cadre la un moment dat. Eliminarea cadrelor reprezintă de fapt încărcarea în fereastra părinte a unui alt document HTML, în locul celui care defineşte cadrele curente. Acest document, ce trebuie înlocuit, este de fapt documentul părinte. Fereastra părinte se accesează din cadrele descendente (figura 5-12) folosind cuvîntul cheie *parent*. Pentru eliminarea cadrelor, spre exemplu, este necesară utilizarea obiectului location al ferestrei părinte, căruia trebuie sa îi atribuim, la *location.href*, o altă adresă Web. Comanda finală va fi:

```
parent.location.href= "http://...";
```

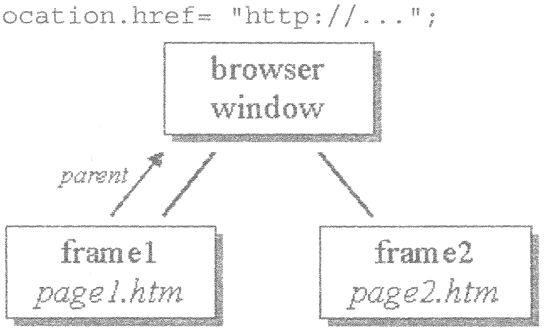


Figura 5-12. Accesarea informaţiei din fereastra părinte în cadrele descendente.

Ultimul caz de acces este acela al utilizării proprietăţilor unui obiect descendent de un script aflat într-un alt obiect descendent, spre exemplu, `page1.htm`. Din figura 5-13 se observă că nu există o legătură directă între cele două obiecte.

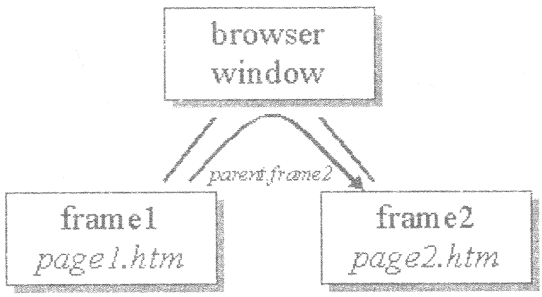


Figura 5-13. Accesarea informației între cadrele descendente.

În concluzie, soluția este utilizarea legăturii amândurora către fereastra părinte:

```
parent.frame2.document.write("Hi, this is frame1
calling.");
```

6. APLICAȚII CGI

Un script CGI (Common Gateway Interface) este un program încărcat pe un server Web şi executat ca răspuns la o cerere a unui utilizator. Un fişier script CGI este scris într-un limbaj de programare astfel încât să fie compilat pentru a rula pe server sau să fie interpretat de un interpretor de pe server.

Printre limbajele utilizate pentru scrierea unor script-uri CGI regăsim:

- limbaje compilate: C, C++, Ada;
- limbaje interpretate: Perl, JCL (Job Control Language: spre exemplu, limbajul de comandă unix sh).

Un script CGI este executat dacă o etichetă de ancorare `<A...>` sau o etichetă imagine `<IMG...>` indică spre un fişier script CGI şi nu către un fişier obişnuit. Statutul acestui fişier ca fiind un script CGI sau doar un fişier HTML este determinat în funcție de poziția fizică a fişierului pe server.

> *Observație: fişierul script este plasat pe aceeaşi maşină pe care rulează server-ul Web, şi nu local, pe maşina adresată.*

În mod obişnuit, fişierele script CGI se găsesc pe server-ul Web, în directorul `cgi-bin`. Localizarea exactă a acestui director pe maşina server este determinată de administratorul Web al respectivului sistem. Localizarea şi controlul directorului `cgi-bin` este determinată de administratorul Web, pentru a preîntâmpina eventuale probleme de securitate ce pot apare dacă programe arbitrare ar avea permisiunea să fie executate de oricine accesează respectiva maşina. Există astfel posibilitatea ca anumiți administratori Web să nu permită crearea unor script-uri CGI pe maşina Web gestionată oricărui tip de utilizator al respectivului sistem.

6.1. Cum lucrează o aplicație CGI?

1. Utilizatorul a introdus informația în forma HTML şi acționează butonul `Submit` (declarat ca element al formei, între etichetele `<FORM>..</FORM>`). Butonul este o legătură la aplicația CGI de pe server.
2. Browser-ul foloseşte metoda `POST/GET` pentru a transmite datele formei către server.

3. Informația este transmisă prin Internet sau Intranet până la serverul Web, care o transferă mai departe aplicației CGI folosind `stdin` al acesteia.
4. Aplicația CGI procesează datele, după care generează răspunsul în format HTML şi îl transmite înapoi spre client (serverul Web trebuie să citească dispozitivul standard de ieşire `stdout` al aplicației CGI).
5. Browser-ul Web interpretează codul HTML recepționat şi afişează rezultatul pe ecranul client. În general, fişierul HTML de răspuns conține notificații asupra procesării datelor.

În mod ideal, un script urmează paşii (folosind rutine corespunzătoare pentru `do-initialize`, `do-process` şi `do-output`):

1. Declarații de început ale programului.
2. Programul apelează `do-initialize`.
3. Programul apelează `do-process`.
4. Programul apelează `do-output`.
5. Sfârşitul programului.

6.2. Apelul unei aplicații CGI

Eticheta HTML care permite executarea scriptului CGI având ca nume de fişier `dynamic_page` pe server-ul `www.mc.com` este:

```
<A HREF="http://www.mc.com/cgi-bin/dynamic_page">
Dynamic page
</A>
```

În momentul în care server-ul Web procesează un apel de fişier, dacă acel fişier se găseşte în directorul `cgi-bin` atunci, în condițiile în care este marcat ca fişier executabil, script-ul va rula pe server. Dacă fişierul nu este executabil, se va genera un mesaj de eroare.

Ca rezultat al executării unui script, este posibilă afişarea unei pagini HTML sau a unei imagini. În plus, script-ul poate accesa în timpul execuției variabile de mediu pentru a utiliza informații auxiliare despre procesul pe care trebuie să îl realizeze. Prima linie a datelor returnate **trebuie** să fie:

Tipul datei returnate	**Text**
Pagină HTML	`Content-type: text/html`
Imagine gif	`Content-type: image/gif`

Un exemplu de script CGI (pe un server Web aflat pe un sistem rulând Unix) care returnează lista utilizatorilor curenți legați pe serverul Web:

```
#!/bin/sh
echo Content-type: text/html
echo
echo "<HTML>"
echo "<HEAD>"
echo "</HEAD>"
echo "<BODY>"
echo "<H2>Users logged in are:</H2>"
echo "<PRE>"
who
echo "</PRE>"
echo "</BODY>"
echo "</HTML>"
```

Observație: sunt necesare ghilimelele în jurul etichetelor (< . . >).

Observație: în sistemul Unix, prima linie este `#!/bin/sh` *iar fișierul trebuie să aibă atributul executabil.*

Principalele variabile de mediu ce pot fi accesate de un script CGI în timpul execuției sunt:

Variabila de mediu	Conținut
`QUERY_STRING`	Informația transmisă către script-ul CGI, prin apelul său (poate fi ieşirea unei forme sau orice alte date generate dinamic sau static).
`REMOTE_ADDR`	Adresa Internet a maşinii gazdă (*host machine*) ce realizează cererea.

În Anexa2 sunt puse la dispoziție codurile sursă ale unor programe C++ de procesare a informației ca aplicații CGI (complilate cu compilator GNU C++).

6.3. Decodificarea datelor transmise către un script CGI

În cazul folosirii unei forme, informația colectată în formă este transmisă catre script-ul CGI pentru prelucrare. Această informație este plasată în variabila de mediu `QUERY_STRING`. În acest caz, pentru transferarea explicită a informației către variabila de mediu `QUERY_STRING`, se utilizează o formă modificată pe baza unei etichete,în care datele sunt adăugate după adresa ce anunță script-ul CGI. Caracterul `?` se utilizează pentru a separa atributul URL şi data transmisă către script. Spre exemplu:

```
<A HREF="/cgi-bin/script?name=Your+name&action=find">
Link</A>
```

Informația `"name=Your+name&action=find"` este plasată în variabila de mediu `QUERY_STRING` iar script-ul CGI `script` este executat.

Un exemplu de aplicație de parsing (de interpretare a datelor transmise prin mecanismul CGI) este utilizată pentru extragerea componentelor individuale din `QUERY_STRING` şi se bazează pe următoarea clasă C++ (vezi Anexa2).

```
class Parse
{
public:
  Parse( char [] );
  ~Parse();
  void set( char [] );
  char *get_item( char [], int pos=1, bool=false
);
  char   *get_item_n(   char   [],   int   pos=1,
bool=false );
protected:
  void remove_escape(char []);
  int hex( char );
  char *map_uname( char [] );
private:
  enum { SEP = '&' }; // Separator de câmpuri
  char *the_str; // şirul interpretat
  int  the_length; // lungimea şirului
};
```

Membrii clasei sunt:

Metoda	Funcția obiectiv
`Parse`	Setează şirul de caractere ce va fi interpretat (parsed).
`set`	Setează un şir de caractere diferit pentru a fi interpretat.
`get_item`	Returnează şirul asociat cuvântului cheie transmis ca parametru. Dacă nu există informație de interpretat, returneaza NULL.
`get_item_n`	Returnează şirul asociat cuvântului cheie transmis ca parametru. Dacă nu există informație de interpretat, returnează şirul nuli (vid).

În cazul folosirii metodelor `get_item` şi `get_item_n`, al doilea parametru (opțional) specifică potrivirile şirului asociat cuvântului cheie pentru returnare. Aceasta permite regăsirea informației

ataşate cuvintelor cheie prestabilite. În plus, şirurile returnate vor fi amendate de următoarele substituții:

- `+`

Va fi convertit în spațiu.

- `=`

Va separa numele câmpului de valoarea câmpului.

- `?`

Va marca începutul datelor din formă în linia de comandă.

- `&`

Va fi folosit ca delimitator între perechile nume/valoare pentru fiecare câmp al formei HTML.

- `%HH`

Va fi convertit în caracterul a cărui valoare hexazecimală este `HH`.

- `~user`

Va fi înlocuit de calea absolută către folderul utilizatorului doar dacă al treilea parametru (opțional) este `true`.

> *Observație: Definiția* `O_MAP` *va conduce la neincluderea pentru procesare a codului pentru* `~username`. *Aceasta este explicația faptului că acest cod poate fi compilat şi pentru maşini care nu suportă funcția sistem* `map_uname` *definită în fişierul header* `cgi/pwd.h`.

Spre exemplu, dacă `QUERY_STRING` conține:

```
tag=one&name=mike&action=%2B10%25&tag=two&log=~ma
s/log&tag=three
```

atunci următorul program (vezi Anexa2), compilat, la rulare:

```
enum bool { false, true };

#include <iostream.h>
#include <stdlib.h>

#include "parse.h"
#include "parse.cpp"

void main()
{
 char *query_str = getenv("QUERY_STRING");

 Parse list( query_str );

cout <<"name ="<<list.get_item_n("name")<<"\n";
cout<<"action="<<list.get_item_n("action")<<"\n";
```

```
cout<<"log="<<list.get_item_n("log",1,true)<<"\n";
 for ( int i=1; i<=4; i++ )
 {
 cout<<"tag("<<i<<")=";
 cout<<list.get_item_n("tag",i)<<"\n";
 }
}
```

va conduce la ieşirea:

```
name   = mike
action= +10%
log    = /usr/staff/mas/log
tag   (1) = one
tag   (2) = two
tag   (3) = three
tag   (4) =
```

6.4. Post versus Get

Până acum, referințele CGI au utilizat ca metodă de transmitere a informației `GET` (informația transferată fiind plasată în variabila de mediu `QUERY_STRING` pentru a fi procesată de script-ul CGI).

O metodă alternativă este `POST`: datele sunt transmise printr-un flux separat şi devin intrarea standard pentru script-ul CGI. Metoda utilizată este specificată în eticheta `<FORM ..>` folosind atributul `METHOD="get"` sau `METHOD="post"`. Metoda implicită este `GET`.

Spre exemplu:

Generated form	HTML markup required

Try it (get)	`<FORM METHOD="get"`
	`ACTION="http://host/cgi-bin/mas_form">`
	`<INPUT TYPE="text" NAME="name"`
	`SIZE=20 VALUE="Try it (get)">`
	`</FORM>`

Try it (post)	`<FORM METHOD="post"`
	`ACTION="http://host/cgi-bin/mas_form">`
	`<INPUT TYPE="text" NAME="name"`
	`SIZE=20 VALUE="Try it (post)">`
	`</FORM>`

Figura 6-1. Un exemplu de script CGI.

La utilizarea atributului POST sunt setate următoarele variabile de mediu:

Variabila de mediu	Conține
CONTENT_LENGTH	Lungimea informației transmise prin intrarea standard către programul CGI.
CONTENT_TYPE	Tipul MIME al datei. MIME (Multipurpose Internet Mail or Messaging Extensions), dezvoltat şi adoptat de Internet Engineering Task Force, permite fişierelor de diferite tipuri să fie transmise prin rețelele ce se bazează pe Transmission Control Protocol/Internet Protocol (TCP/IP). Acest protocol, actualmente o extindere a Simple Mail Transport Protocol (SMTP), acoperă domeniul datelor binare, audio şi video, schimbului electronic de informație (Electronic Data Interchange EDI) etc. În timp ce SMTP cunoaşte cum să transfere doar conținutul unui singur fişier bazat pe caractere ASCII (American Standard Code for Information Interchange), MIME descrie o modalitate de codificare a antetului şi tuturor componentelor ataşate, astfel încât să fie identificate de sistem ca un singur fişier. Un obiect MIME este apoi transferat printr-un mesaj SMTP.

Un exemplu de script pentru înregistrarea datelor transmise de utilizator într-un fişier log este redat în figura 6-2.

Generated form	HTML markup required
fill in	<FORM METHOD="post" ACTION="http:/machine/cgi-bin/mas_cgi1"> <INPUT TYPE="text" NAME="name" SIZE=20 VALUE="fill in">

Figura 6-2. Script pentru înregistrarea datelor utilizatorului.

6.5 Verificarea script-urilor CGI

Dacă la apelul unui script CGI există probleme de rulare, atunci trebuie verificat, în momentul implementării, dacă script-ul CGI este:

- Executabil;
- Plasat în directorul `cgi-bin`. Localizarea acestui director este definită de administratorul Web-server;
- Capabil să acceseze toate fişierele necesare (deoarece un script rulează în directorul `cgi-bin` al serverului, orice cale relativă către fişiere trebuie să fie relativă la acest director.

- La returnarea unei pagini HTML care accesează imagini sau foloseşte legaturi hypertext, este preferabilă specificarea acestora prin forma `http` a URL.

- Pe un server Unix trebuie setat bitul `set uid` al fişierului script executabil. Aceasta va conduce la executarea programului ca şi cum ar fi rulat de proprietarul fişierului (aspect necesar în cazul în care programul trebuie să acceseze resurse pe care doar creatorul script-ului le poate accesa, dar care poate pune problema unor distorsiuni de securitate).

7. APLICAȚII JAVA PENTRU PAGINI WEB

7.1. Noțiuni introductive

Java permite crearea codurilor executabile de mici dimensiuni (applet-uri), care să ruleze în spațiul de memorie al unui browser. Ca limbaj complet de programare, Java permite acțiuni diverse, ca desenarea, completarea bitmap, controlul şi răspunsul la evenimentele utilizator (user events), încărcarea documentelor de la adresa specificată etc. Din observațiile realizate asupra applet-urilor Java folosite astăzi pe Web, putem semnaliza că dezvoltatorii de Web folosesc Java în patru direcții principale:

- crearea graficii animate, incluzând secvențe repetate de bitmap, prelucrarea atributelor bitmap, desene animate.
- crearea obiectelor grafice (grafice, diagrame) pentru care nu este necesară preluarea datelor de pe server prin script CGI/ASP. În mod uzual, un grafic se crează printr-o interogare a unei baze de date, urmată de producerea unui GIF şi trimiterea acestuia la browser. Această strategie are dezavantajul timpului mărit pentru preluarea fişierelor GIF (în general de dimensiuni mari) şi pentru crearea unui GIF pe server.
- crearea "controalelor": butoane, egalizoare grafice etc. Prin definirea şi preluarea evenimentelor folosind Java applet, programatorul poate crea obiecte grafice de control.
- crearea aplicațiilor bazate pe colecții de controale ca zone de editare, butoane, spații de selecție etc. Această capabilitate este similară aplicațiilor JavaScript, dar programul este deja inclus şi compilat, deci nu se dezvăluie codul sursă.

Notă: Java este un limbaj de programare orientat obiect derivat din C++, având toate avantajele POO ale C++ şi evitând unele aspecte neplăcute (alocare dinamică de memorie) pentru mărirea robusteții şi securității codului. Java pune la dispoziție o colecție bogată de biblioteci (*packages*), ce extind capabilitățile limbajului. Printre acestea se numără o bibliotecă pentru interfața utilizator (AWT), o bibliotecă I/O, o bibliotecă pentru lucrul în rețea etc.

Programarea Java poate fi folosită pentru crearea applet-urilor ce se încarcă pe Web şi sunt executate sub un browser, sau pentru crearea aplicațiilor autonome (stand-alone). Subiectul acestui capitol este crearea şi folosirea applet-urilor sub browser de navigare Web.

7.2. Primul exemplu: appletul *Hello World!*

Să reluăm binecunoscutul exemplu introductiv `Hello World!` într-o noua variantă:

```
//Hello.java
  import java.awt.*;
  import java.applet.*;
  public class Hello extends Applet
  {
   public void paint(Graphics g)
   {
    for (int x = 0 ; x < size().height ; x++)
    {
     g.drawString("Hello World!", 10, x*15);
    }
   }
  }
```

Salvați fișierul cu numele `Hello.java` şi compilați folosind linia de comandă:

```
javac -g Hello.java
```

Fișierul `Hello.class` conține codul binar al applet-ului. Putem să definim acum conceptul de **applet**: codul creat de compilator, un fișier binar ce conține codul executabil. Pentru a rula acest cod executabil, trebuie încărcat într-un browser cu capabilități Java.

Aceasta se poate realiza folosind următorul document HTML, salvat în acelaşi director cu applet-ul:

```
<title>Hello World Demo</title>
<hr>
<applet code=Hello.class width=300 height=120>
</applet>
<hr>
```

Încărcați documentul HTML în browser-ul Netscape Navigator şi obțineți applet-ul Hello World! rulând într-o mică fereastră 300x120 din pagina HTML (figura 7-1).

Vom parcurge programul linie cu linie, pentru a înțelege cum lucrează. Codul a fost, pentru acest scop, numerotat:

```
//Hello.java
1 import java.awt.*;
2 import java.applet.*;
3 public class Hello extends Applet
4 {
5   public void paint(Graphics g)
6   {
```

```
7       for (int x = 0 ; x < size().height ; x++)
8       {
9          g.drawString("Hello World!", 10, x*15);
10      }
11   }
12 }
```

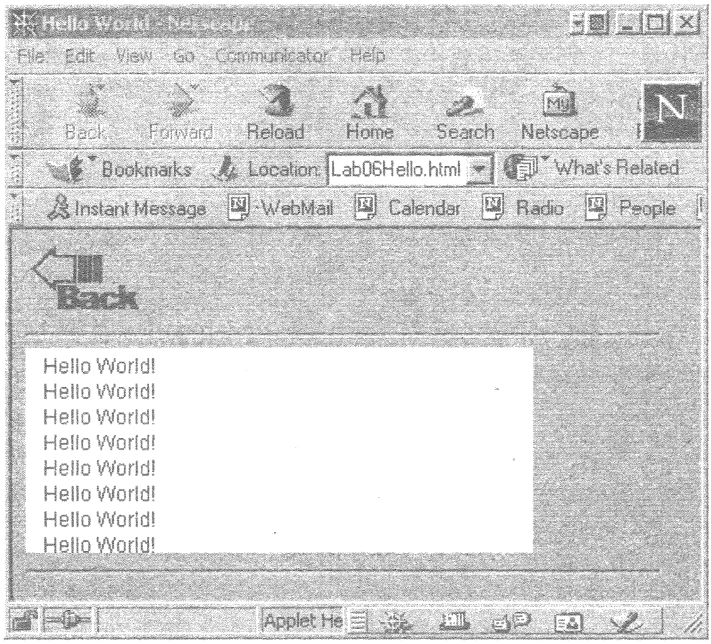


Figura 7-1. Applet-ul Hello World!

Primele două linii importă bibliotecile necesare, analog liniilor *include* din C sau C++. În Java, bibliotecile sunt organizate ierarhic (`java libraries`), putând fi încărcate doar piesele necesare. În cazul exemplului, sunt încărcate doar piesele `AWT` şi `applet`. În linia 3 se declară o nouă clasă numită `Hello`. Numele fişierului şi clasei trebuie să fie analoge (pentru appletul `Hello`, fişierul trebuie să fie `Hello.java`). Clasa este derivată din `Applet class` şi o extinde. Clasa are o metodă numită `paint` (provenită din superclasa `Component` a pachetului `Applet`). Ea primeşte o referinţă la clasa `Graphics`. Clasa conţine funcţii de desenare a liniilor, cercurilor, dreptunghiurilor etc. (temă: încercaţi să folosiţi unele dintre aceste metode prin simplul apel în metoda `paint`).

Ca exemplu, codul următor demonstrează cum se setează culoarea şi tipul caracterului:

```
import java.awt.*;
import java.applet.*;
```

```
public class Hello extends Applet
{
 public void paint(Graphics g)
 {
  g.setColor(Color.red);
  g.setFont(new Font("Helvetica",Font.PLAIN,24));
  for (int x = 0 ; x < size().height ; x++)
  {
   g.drawString("Hello World!", 10, x*15);
  }
 }
}
```

Observație: pentru fiecare nouă rulare a modificărilor în pagina HTML consacrată applet-ului Hello World! folosiți butonul `Reload`. În cazul în care nu obțineți rezultatul scontat, deschideți o nouă fereastră Netscape!

7.3. Biblioteca AWT

Biblioteca AWT permite crearea interfețelor utilizator în interiorul applet-urilor Java. Biblioteca conține controalele clasice pentru dezvoltarea unei interfețe simple: butoane, spații de editare (edit areas), spații de marcare (check boxes) etc. În plus, conține câteva containere diferite, utile pentru aranjarea controalelor pe ecran.

Exemplul următor demonstrează cum să creați un buton (push button) şi să-l plasați într-un container numit `BorderLayout`:

```
/ ButtonTest.java
import java.awt.*;
import java.applet.*;
public class ButtonTest extends Applet {
 int y;
 Button b;
 public ButtonTest()  {
  y = 10;
  setLayout(new BorderLayout());
  b = new Button("Apply");
  add("North", b);
 }
 public void paint(Graphics g) {
  for (int x = 0 ; x < size().height ; x+=10) {
   g.drawString("Hello World!", y, x);
  }
 }
}
```

Acesta nu este foarte diferit de programul `Hello World!` inițial. Diferența constă în constructorul applet-ului, unde se inițializează variabila `y`, se setează un layout pentru applet, se crează un control de tip buton care afișează cuvântul "Apply" și care adaugă apoi butonul în layout. Un `BorderLayout` are cinci puncte de atașare: `North`, `South`, `East`, `West` și `Center`. Rezultatul depinde însă și de dimensiunea applet-ului pe ecran. În exemplu, butonul este plasat la `North` (sus). Restul ferestrei conține textul generat cu metoda `paint()`.

Compilați și rulați acest program folosind pașii descriși anterior. Studiați documentația pentru programarea controalelor și containerului `BorderLayout`, ca și a metodei `setLayout` și încercați să adăugați metode în clasa declarată inițial (a applet-ului). Încercați să plasați și alte controale în locul (nu adăugând!) controlului de tip buton.

7.4. Preluarea evenimentelor

La clic pe butonul din exemplul anterior nu se întâmplă nimic. Pentru a face să se întâmple ceva, este necesar să controlați respectivul eveniment. Cel mai ușor mod este acela de a adăuga o funcție de control al evenimentului la clasa applet-ului. Un alt mod este acela de a moșteni un nou buton din butonul de bază și de a controla evenimentul în noua clasă.

Exemplul următor evidențiază manipularea unui extrem de simplu eveniment:

```
// ButtonTest.java
import java.awt.*;
import java.applet.*;
public class ButtonTest extends Applet
{
   int y;
   Button b;
   public ButtonTest()
   {  y = 10;
      setLayout(new BorderLayout());
      b = new Button("Apply");
      add("North", b);
   }
   public void paint(Graphics g)
   {for (int x = 0 ; x < size().height ; x+=10)
      {g.drawString("Hello World!", y, x);}
   }
   public boolean action(Event ev, Object arg)
   {
```

```
        if (ev.target instanceof Button)
        {
            y+=10;
            repaint();
            return true;
        }
        return false;
    }
}
```

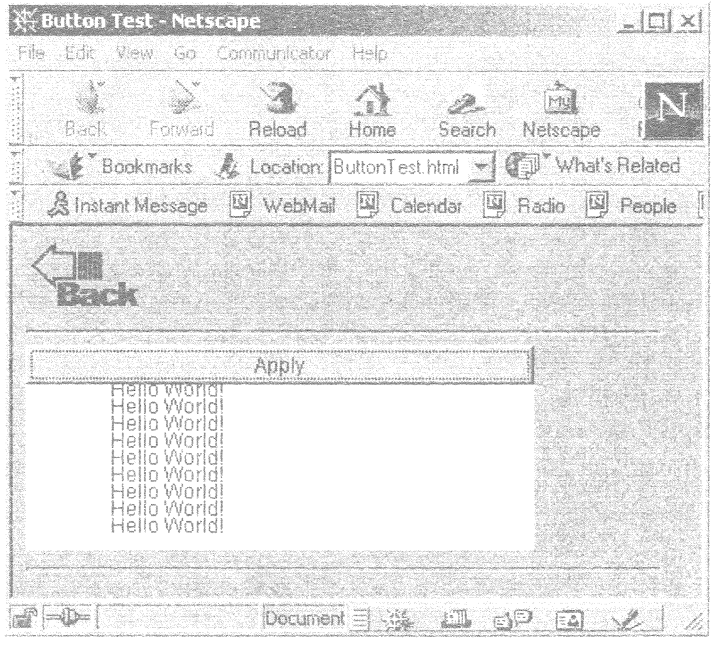


Figura 7-2. Completarea răspunsului butoanelor într-un applet.

Metoda `action`, în acest applet, abordează şirul de evenimente, examinând, pe măsură ce evenimentele sosesc, dacă acestea au ca sursă obiectul `Button`. Dacă răspunsul este afirmativ, se incrementează `y` şi forțează applet-ul să facă `repaint`. La rularea programului veți observa deplasarea coloanei `Hello World` cu 10 pixeli la fiecare apăsare de buton.

Dacă applet-ul ar conține două butoane, unul la nord şi celălalt la sud, şi am dori să răspundă diferit, codul anterior va avea probleme, deoarece orice eveniment de la orice buton va lansa acțiunea. Răspunsul specific fiecărui buton (figura 7-2) se va realiza prin adăugarea în cod a următoarelor secvențe:

```
public boolean action(Event ev, Object arg)
{
 if (ev.target instanceof Button)
 {
  if("Apply".equals(ev.arg))
  {
   y+=10;
   repaint();
   return true;
  }
 }
 return false;
}
```

Atributul `ev.arg` aduce eticheta butonului apăsat; prin compararea acestuia cu componentele listei etichetelor de butoane se poate determina butonul apăsat. Bineînțeles, se pot adăuga şi alte butoane la `BorderLayout` (la S, E şi V), personalizând evenimentele care să conducă la afişări.

7.5. Panouri

Din exemplul precedent trebuie reținute două aspecte importante:

A. Putem avea un singur control în fiecare din cele cinci poziții ale `BorderLayout`.

B. Dacă plasăm butoane în afara colțurilor `BorderLayout`, desenarea nu este o intrare separată. Spre exemplu, şirul "`Hello World!`" este afişat pe suprafața applet-ului iar butoanele acoperă părți ale acesteia; prin urmare trebuie controlată afişarea în suprafața corespunzătoare.

Pentru a rezolva ambele probleme, se poate folosi un **panou** (`panel`) sau, în cazul unei suprafețe de afişare-desenare, o **pânză** (`canvas`). Un `panel` este preferabil pentru colectarea împreună a seturilor de controale, aranjându-le într-un rând care controlează dimensiunile, dacă limitările de spațiu ale panoului o fac necesară:

```
// SimpleUI.java
import java.awt.*;
import java.applet.*;
class PanelGraph extends Panel
{
   int x = 10;
   public void paint(Graphics g)
   {
    for (int y = 0 ; y < size().height ; y+=15)
```

```
        {
          g.drawString("Hello World!", x, y);
        }
    }
    public void changeX()
    {
        x += 10;
        repaint();
    }
}

class PanelUI extends Panel {
    Choice c;
    PanelGraph pg;
    public PanelUI(PanelGraph pgIn)
    {
        add(new Button("Apply"));
        add(new Label("Test Colors", Label.RIGHT));
        add(c = new Choice());
        c.addItem("Red");
        c.addItem("Green");
        c.addItem("Blue");
        pg = pgIn;
    }
    public boolean action(Event ev, Object arg)
    {
        if (ev.target instanceof Button)
        {
              pg.changeX();
              return true;
        }
        return false;
    }
}
public class SimpleUI extends Applet
{
    PanelGraph pg;
    PanelUI pui;
    public SimpleUI()
    {
        setLayout(new BorderLayout());
        pg = new PanelGraph();
        add("Center", pg);
        Panel p = new PanelUI(pg);
        add("North", p);
    }
}
```

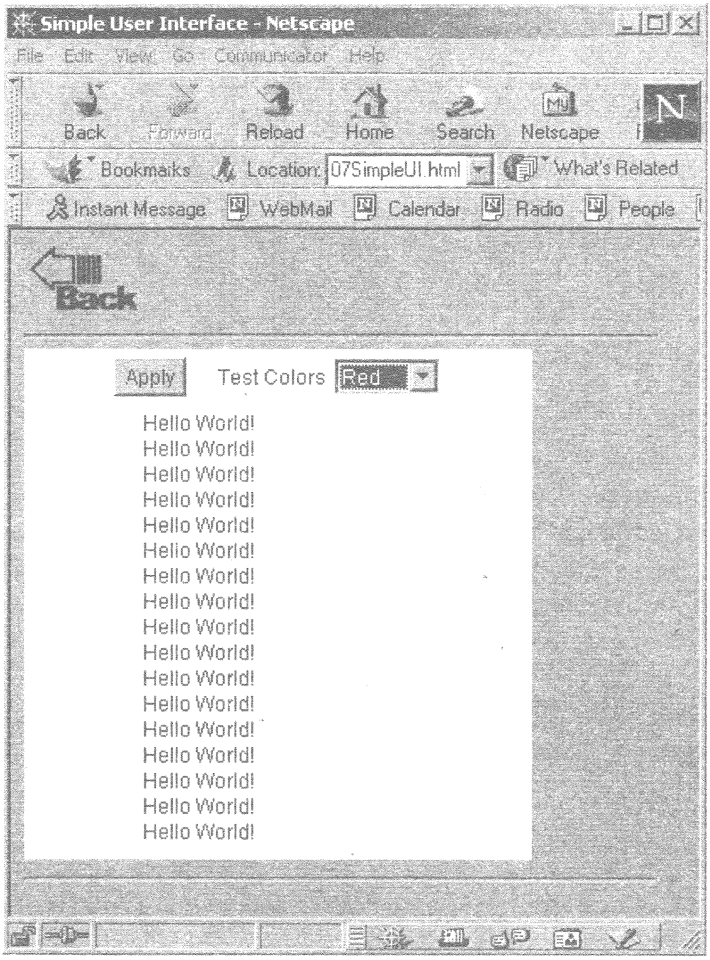


Figura 7-3. Un applet ce gestionează un buton, o etichetă şi o listă de selecție.

În acest program sunt două noi clase, care extind clasa `Panel`. `PanelGraph` consideră panoul ca o suprafață de afişare, în care se scrie "`Hello World`". Această clasă are o metodă de desenare identică celei anterioare, cu excepția metodei `changeX`, apelabilă de o altă clasă, pentru deplasarea şirului de caractere "`Hello World`". Şi clasa `PanelUI` extinde `Panel`: constructorul său adaugă trei controale pe panou (un buton, o etichetă-`label` şi o listă de selecție, care însă nu realizează nimic deocamdată). Panoul recunoaşte evenimentele de apăsare a butoanelor, la care reacționează prin apelul metodei `changeX` din clasa `PanelGraph`. Clasa `Applet` crează cele două panouri şi le adaugă la `layout`, restul acțiunilor fiind realizate de `panels`.

Panoul va conține cele trei controale la nord, restul ferestrei fiind alocată panoului grafic (figura 7-3). La clic pe buton, textul se va deplasa.

Pentru a completa programul, vom modifica în `PanelGraph` culoarea, în funcție de culoarea selectată de utilizator (figura 7-4). Clasa `PanelGraph` va fi modificată astfel:

```
class PanelGraph extends Panel
{
   int x = 10;
   Color c = Color.red;
   public void paint(Graphics g)
   {
    g.setColor(c);
    for (int y = 0 ; y < size().height ; y+=15)
      {
            g.drawString("Hello World!", x, y);
      }
   }
   public void changeX()
   {
      x += 10;
      repaint();
   }

   public void changeColor(Color cin)
   {
      c = cin;
      repaint();
   }
}
```

Vom modifica şi clasa PanelUI astfel:

```
public boolean action(Event ev, Object arg)
   {
      if (ev.target instanceof Button)
      {
         if ("Red".equals(c.getSelectedItem()))
           pg.changeColor(Color.red);
         if ("Green".equals(c.getSelectedItem()))
           pg.changeColor(Color.green);
         if ("Blue".equals(c.getSelectedItem()))
           pg.changeColor(Color.blue);
         return true;
      }
      return false;
   }
```

Figura 7-4. Un applet pentru selecția culorii din lista de culori.

7.6. Parametrii unui applet

Există multe cazuri în care se doreşte personalizarea applet-urilor prin transmiterea parametrilor la apelul său. Aceasta se realizează prin transmiterea dinspre HTML a parametrilor dorițI:

```
<title>Hello World Demo</title>
<hr>
<applet code=Hello.class width=300 height=120>
<param name=rows value="4">
</applet>
<hr>
```

Parametrii pot avea orice nume dorim. Valoarea parametrilor transmişi dinspre HTML se preia cu metoda `getParameter` a clasei `Applet`:

```
String r = getParameter("rows");
```

Dacă numele parametrului nu există, atunci `getParameter` returnează `null`:

```
if (r == null)
{
  = 10; // default value
}
else
{
 r = Integer.parseInt(rs);
}
```

7.7. Imagini şi animaţie

Java permite programarea multi-tasking şi facilitează lucrul cu imagini (bitmaps). Aceste două facilităţi pot fi combinate pentru a crea animaţie. Codul de mai jos este un exemplu de simplă animaţie (figura 7-5): o imagine este deplasată pe verticală, de sus în jos.

```
import java.awt.*;
import java.applet.*;
public class Logo extends Applet implements
Runnable
    {
    Image img;
    Thread thd = null;
    int i;
    int imgWidth = 273;
    int imgHeight = 202;
    public void run()
    {
       img=getImage(getCodeBase(), "tim3.gif");
       if (img != null)
       {
        i=imgHeight;
        repaint();
        while (true)
        {
         try{Thread.sleep(1000);}
         catch(InterruptedException e){}
                i=0;
                while (i<imgHeight)
```

```
                {
                 repaint();
                 try{Thread.sleep(50);}
                 catch (InterruptedException e){}
                    i+=4;
                }
            }
        }
    }

    public void update(Graphics g)
    {
        if (img != null)
        {
         g.clipRect(0, 0, imgWidth, i);
         g.drawImage(img,0,i-imgHeight, null);
        }
    }

    public void start()
    {
        if (thd == null)
        {
            thd = new Thread(this);
            thd.start();
        }
    }
    public void stop()
    {
        thd = null;
    }
}
```

Pentru înțelegerea codului, trebuie parcursă documentația Sun JDK despre clasa `thread`. Metoda `run` reprezintă un nou fir de execuție (`thread`). În acest fir separat, codul încarcă o imagine și o animează prin redesenare într-o nouă poziție la fiecare 50 milisecunde, într-o secvență `while` condiționată în interior de dimensiunea zonei de afișare. Apoi, codul așteaptă timp de o secundă și repetă animația.

Pentru desenarea imaginii într-o nouă poziție, se apelează metoda `repaint`. Această funcție apelează metoda supradefinită `update`, similară cu metoda `paint`, cu excepția faptului că nu șterge fereastra înainte de desenare.

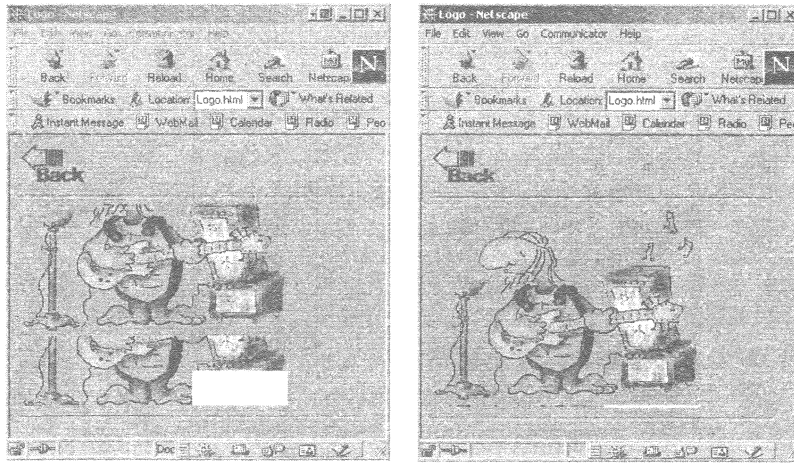


Figura 7-5. Deplasarea unei imagini pe verticală.

Metoda `update` afişează desenul dat la poziția specificată de indexul `i`. În cazul în care se doreşte animarea unui set de `N` cadre (`N` este o constantă, spre exemplu 15), trebuie doar să fie incărcate cele `N` cadre (`frames`) într-un vector de imagini şi să le desenăm pe fiecare (prin apeluri ale metodei `paint`), secvențial, în metoda `update`.

7.8. Câmpuri text

Componentele standard ale unei interfețe sunt special realizate pentru a oferi sau a prelua informații de la utilizator:

`Button`	Buton de validare
`Canvas`	Componentă generică pentru crearea unui obiect "`canvas`"
`Checkbox`	Casetă de validare"
`Label`	Un rând de text de tip "`label`"
`List`	Lista de selecție
`Scrollbar`	Bară de derulare
`TextField`	Câmp pentru introducerea unei linii de text
`TextArea`	Câmp pentru editarea unor linii multiple de text

Componenta **`TextField`** permite introducerea de către utilizator, în cadrul unei cutii ("`box`") a unei singure linii de text. Numărul de caractere care sunt vizibile în rândul de text poate fi specificat explicit.

Adăugarea unei componente `TextField` într-o fereastră se face asemănător cu adăugarea unui buton sau a oricărei alte componente:

```
TextField t = new TextField(12);
add(t);
```

Parametrul 12 din apelul constructorului `TextField(12)` setează lungimea textului care urmează să fie afişat la aproximativ 12 caractere. Utilizatorul poate tasta mai mult de 12 caractere, dar numai unele dintre ele vor fi afişate.

Observatie: numărul de caractere afişate depinde de caracterele tastate. Se pot afişa mai multe caractere, dacă forma caracterelor este mai îngustă.

Dacă un şir este specificat ca parametru:

```
TextField t = TextField("your name");
```

atunci câmpul text este creat ca având implicit valoarea parametrului. Se pot specifica în apel atât textul implicit, cât şi lungimea textului, astfel:

```
TextField t = new TextField("your name", 20);
```

Dacă nu este setată o valoare implicită, deci nu se specifică lungimea printr-un parametru, rezultatul este un şir text de un singur caracter. Un exemplu simplu de introducere a unui control al clasei `TextField` într-un applet este următorul:

```
import java.awt.*;
import java.applet.*;
public class Applet1 extends Applet{
    TextField text;
    public void init(){
       text = new TextField(20);
       add(text);
    }
}
```

În primele două linii ale exemplului anterior se specifică faptul că programul foloseşte clase din pachetele `AWT` si `APPLET`. Instrucțiunea: `public class Applet1 extends Applet` arată că `Applet1` derivă din clasa `Applet`.

În continuare se declară "`text`" ca fiind un obiect al clasei `TextField`. Apoi se apelează metoda `init()` a clasei `Applet`. În cadrul metodei, se crează un nou obiect `TextField`:

```
text=new TextField(20);
```

Prin linia `add(text)` se adaugă obiectul `TextField` applet-ului.

Pentru a rula acest applet, se salvează cu numele `Applet1.java`, se compilează, apoi se crează un document `.html` astfel:

```
<html>
<head>
<title>Applet Test Page</title>
</head>
<body>
<h1>Applet Test Page</h1>
<applet code="Applet1.class" width=200 height=150
        name="Applet1">
</applet>
</body>
</html>
```

Câmpurile de tip **`TextArea`** sunt asemănătoare cu `TextField`, numai că permit introducerea mai multor linii de text. Numărul de linii şi numărul de caractere vizibile de pe o linie din cadrul ariei de text pot fi configurate. Exemplu:

```
TextArea t = new TextArea(5,20);
add(t);
```

Parametrii 5 şi 20 ai constructorului specifică numărul de linii şi numărul de caractere de pe o linie. Daca textul introdus depăşeşte marginea ariei de afişat, apare o bară derulantă care permite utilizatorului să se deplaseze între diferitele secțiuni ale ferestrei.

Dacă se specifică un şir ca parametru:

```
TextArea t=TextArea("Şir de caractere",7, 25);
```

atunci aria de text este creată având şirul parametru text implicit, încărcat la început. Pentru a introduce o parte a acestui şir la începutul liniei următoare se foloseşte caracterul "\n" astfel:

```
TextArea t=TextArea("Şir de \n ",7,25);
```

Să studiem modul de utilizare a unui text introdus printr-un `TextField` în exemplul următor:

```
import java.awt.*;
import java.applet.*;
public class Applet2 extends Applet{
   TextField textField;

   public void init(){
        textField = new TextField(20);
        add(textField);
    }

   public void paint(Graphics g){
```

```
        String s = textField.getText();
        g.drawString(s, 40, 50);
    }

    public boolean action(Event event,Object arg){
        repaint();
        return true;
    }
}
```

Se observă în codul anterior clasa `Applet2`, care derivă din clasa `Applet1`. În plus față de exemplul precedent, se apelează metoda `paint()` a clasei `Applet`. Prin linia:

```
String s=textField.getText();
```

se preia un şir introdus cu `TextField`, se afişează textul în cadrul suprafeței definite a applet-ului:

```
g.drawString(s,40,50);
```

apoi se apelează metoda `action()` pentru a forța redesenarea suprafatei de afişare. Fisierul `.html` necesar pentru includerea şi rularea applet-ului va semăna cu cel anterior, cu deosebirea că se înlocuieşte `Applet1` cu `Applet2`.

Un exemplu care integrează câteva dintre obiectele discutate în acest capitol: `Integrare.java`:

```
import java.awt.*;

class integrare extends Frame {
Label lab1=new Label("Marginea inf. a int.:");
Label lab2=new Label("Marginea sup.:");
Label lab3=new Label("Numarul de pasi (de integrare)");
Label lab4=new Label("Introducere corecta?");
Label lab5=new Label("Rez=");
Label infLabel=new Label();
Label supLabel=new Label();
Label nrpLabel=new Label();
TextField tf1=new TextField("text");
TextField tf2=new TextField("3");
TextField tf3=new TextField("8");
Button but1=new Button("Da");

public static void main(String args[]) {
 integrare p=new integrare();
}

public integrare(){//constructor
 super("Calcularea unei integrale");
//titlul ferestrei
 setLayout(new GridLayout(5,3,0,10));
 addComponents();//metoda definita mai jos
 pack();//aranjeaza in pagina conform grilei
```

```
 resize(600,300);
//dimensiunile initiale ale ferestrei
 show();
}

void addComponents() {
 add(lab1);
 add("Center",tf1);
 add(infLabel);
 add(lab2);
 add(tf2);
 add(supLabel);
 add(lab3);
 add(tf3);
 add(nrpLabel);
 add(lab4);
 add(but1);
 add(lab5);
}

public boolean handleEvent(Event e) {
 if(e.id==Event.WINDOW_DESTROY) {
  System.exit(0);
  return true;
 }
 else
if(e.id==Event.ACTION_EVENT&&e.target instanceof Button){
   try {
double inf=(new Double(tf1.getText())).doubleValue();
infLabel.setText(""+inf);
double sup = (new Double(tf2.getText())).doubleValue();
supLabel.setText(""+sup);
int nrPasi = Integer.parseInt(tf3.getText());
nrpLabel.setText(""+nrPasi);
double result = integrTrap(inf,sup,nrPasi);
lab5.setText(" Rezultatul integrarii este:"+result);
return true;
}
catch(NumberFormatException e1) {
lab5.setText("Format gresit la datele de intrare"+e1);
}
catch(Exception e2) {
 lab5.setText(e2.getMessage());}
 return true;
} else
if(e.id==Event.ACTION_EVENT&&e.target
                  instanceof TextField){
                    return true;
                  }
                  else  return false;
       }

double integrTrap(double inf,double sup,int nrPasi) throws
Exception {
if(nrPasi<=0)   throw   (new   Exception("Numar   de   pasi
incorect"));
```

```
else {
double h=(sup-inf)/nrPasi;
if(inf==sup) return 0;
double suma=0;
for(int i=1;i<nrPasi;i++)
  suma+=func(inf+i*h);
return ((func(inf) +func(sup))/2+suma)*h;
}
}

double func(double x) {
  return Math.sqrt(1.2 + Math.sin(2*x+1)*Math.cos(x));
 }
}
```

Programul crează o grilă de obiecte cu ajutorul clasei `GridLayout`. Grila este completată de la stânga la dreapta şi de sus în jos. Fiecare dintre obiectele grilei sunt instanțe ale unor obiecte definite în Java. Prin `event handler` se tratează evenimentul generat de metoda `handleEvent()`. Evenimentul poate fi închiderea ferestrei, apăsarea butonului sau introducerea unui şir de caractere. În cazul în care este apelat obiectul buton, dacă datele sunt corecte, se calculează integrala funcției `func`; iar dacă nu, se afişează un mesaj (figura 7-6).

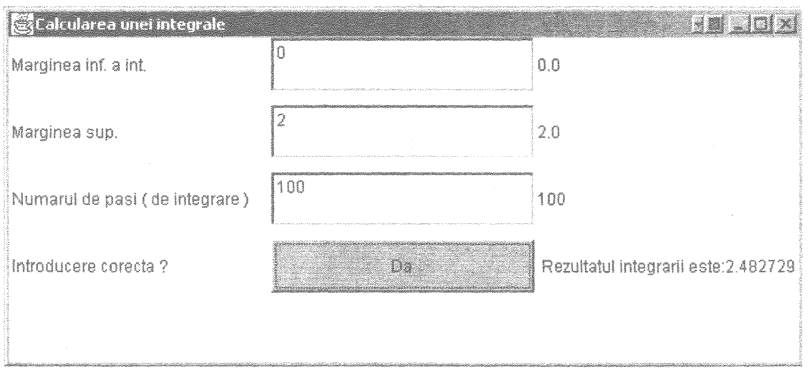


Figura 7-6. Gestionarea mai multor controale Java printr-o grilă de obiecte grafice.

ANEXA 1: LISTA PRINCIPALELOR ETICHETE HTML4.0

Lista simbolurilor folosite:

- URL: adresa unui fişier extern (sau doar numele fişierului, dacă se află în acelaşi director cu documentul HTML);
- ? se înlocuieşte cu număr (spre exemplu: <H?> înseamnă <H1>, <H2>, <H3> etc.);
- % se înlocuieşte cu procent (spre exemplu: <hr align=left width="%"> înseamnă <hr align=left width="50%">);
- *** se înlocuieşte cu text (spre exemplu: ALT="***") propune înlocuirea sevenței cu un text;
- $$$$$$ se înlocuieşte cu număr hexa (spre exemplu: BGCOLOR="#$$$$$$" înseamnă BGCOLOR="#00FF1C");
- ,,, înseamnă separare prin virgulă (spre exemplu: COORDS=",,," înseamnă COORDS="0,0,50,50");
- | se înlocuieşte cu opţiune (spre exemplu: ALIGN=LEFT|RIGHT|CENTER propune folosirea uneia din aceste valori);
- [*] înseamnă legătură către o observație;

ELEMENTE DE BAZĂ

Tip document	<HTML></HTML>	(început şi sfârşit de fişier)
Titlu	<TITLE></TITLE>	(este inclus în header)
Header	<HEAD></HEAD>	(informații descriptive)
Corp	<BODY></BODY>	(corp de pagină)

DEFINIȚIE STRUCTURALĂ

Mărime de caracter	<H?></H?>	(defineşte şase nivele)
Mărime aliniere cu	<H? ALIGN=LEFT|CENTER|RIGHT></H?>	
Divizare	<DIV></DIV>	

Divizare cu aliniere	<DIV ALIGN=LEFT|RIGHT|CENTER></DIV>	
Citat	<BLOCKQUOTE></BLOCKQUOTE>	(de obicei deplasat)
Evidențiere	<EM></EM>	(de obicei aplecat)
Evidențiere prin îngroşare	<STRONG></STRONG>	(de obicei îngroşat)
Citat	<CITE></CITE>	(de obicei aplecat)
Cod	<CODE></CODE>	(listarea codului sursă)
Exemplu de ieşire	<SAMP></SAMP>	
Exemplu de intrare de la tastatură	<KBD></KBD>	
Variabilă	<VAR></VAR>	
Definiție	<DFN></DFN>	(nu prea răspândit)
Adresă de autor	<ADDRESS></ADDRESS>	
Font mare	<BIG></BIG>	
Font mic	<SMALL></SMALL>	

FORMAT DE PREZENTARE

Îngroşat	<B></B>	
Aplecat	<I></I>	
Subliniat	<U></U>	
Evidențiat	<STRIKE></STRIKE>	(nu prea răspândit)
Evidențiat	<S></S>	(nu prea răspândit)

Subscript	`<SUB></SUB>`	
Superscript	`<SUP></SUP>`	
Maşină de scris	`<TT></TT>`	(foloseşte font tipografic)
Preformatare	`<PRE></PRE>`	(nu se ignoră spaţiile şi liniile noi)
Dimensiune	`<PRE WIDTH=?></PRE>`	(în caractere)
Centru	`<CENTER></CENTER>`	(pentru text şi imagini)
Clipire	`<BLINK></BLINK>`	(cel mai ocolit tag)
Mărimea fontului	`<FONT SIZE=?></FONT>`	(de la 1 la 7)
Schimbarea mărimii fontului	`<FONT SIZE="+|-?"></FONT>`	
Mărime de bază a fontului	`<BASEFONT SIZE=?>`	(de la 1 la 7; implicit 3)
Culoarea fontului	`<FONT COLOR="#$$$$$$"></FONT>`	
Selecţia fontului	`<FONT FACE="***"></FONT>`	
Text pe mai multe coloane	`<MULTICOL COLS=?>` `</MULTICOL>`	
Distanţa între coloane	`<MULTICOL GUTTER=?></MULTICOL>`	(implicit 10 pixeli)
Dimensiune coloană	`<MULTICOL WIDTH=?>` `</MULTICOL>`	
Distanţare	`<SPACER>`	
Tip de distanţare	`<SPACER` `TYPE=horizontal|vertical|block>`	

Mărimea distanțării	`<SPACER SIZE=?>`	
Dimensiuni de distanțare	`<SPACER WIDTH=? HEIGHT=?>`	
Aliniere	`<SPACER ALIGN=left|right|center>`	

LINK-URI ŞI FIŞIERE GRAFICE

Link către alt document	<A HREF="URL"></A>	
Link către un target	<A HREF="URL#****"></A>	(în alt document)
	<A HREF="#****"></A>	(în acelaşi document)
Fereastra țintă	<A HREF="URL" TARGET="****||_blank|_self|_parent|_top"></A>	
Definire target în document	<A NAME="****"></A>	
Relație	<A REL="****"></A>	(nu prea răspândit)
Relație inversă	<A REV="****"></A>	(nu prea răspândit)
Încărcare imagine	<IMG SRC="URL">	
Aliniere	<IMG SRC="URL" ALIGN=TOP|BOTTOM|MIDDLE|LEFT|RIGHT>	
Aliniere	<IMG SRC="URL" ALIGN=TEXTTOP| ABSMIDDLE|BASELINE|ABSBOTTOM>	
Alternare text	<IMG SRC="URL" ALT="****">	(dacă imaginea nu este încărcată)
Mapare imagine	<IMG SRC="URL" ISMAP>	(cere un CGI)
Utilizare imagine mapată	<IMG SRC="URL" USEMAP="URL">	

Definire mapare	<MAP NAME="****"></MAP>	
Definire regiune mapată	<AREA SHAPE="RECT" COORDS=",,," HREF="URL"|NOHREF>	
Dimensiuni	<IMG SRC="URL" WIDTH=? HEIGHT=?>	(în pixeli)
Chenar	<IMG SRC="URL" BORDER=?>	(în pixeli)
Spațiu inconjurator	<IMG SRC="URL" HSPACE=? VSPACE=?>	(în pixeli)
Browser cu rezoluție mică	<IMG SRC="URL" LOWSRC="URL">	
Forțare client	<META HTTP-EQUIV="Refresh" CONTENT="?; URL="URL">	
Inserare obiect	<EMBED SRC="URL">	(inserare obiect în pagină)
Dimensiuni obiect	<EMBED SRC="URL" WIDTH=? HEIGHT=?>	

DIVIZORI

Paragraf	<P></P>	(nu e necesară închiderea tag-ului)
Aliniere text	<P ALIGN=LEFT|CENTER|RIGHT> </P>	
Linie nouă (CR: carriage return)	 	(un singur CR)
Elimină spațiul în jurul textului	<BR CLEAR=LEFT|RIGHT|ALL>	
Linie orizontală	<HR>	
Aliniere	<HR ALIGN=LEFT|RIGHT|CENTER>	
Grosime	<HR SIZE=?>	(în pixeli)
Lungime	<HR WIDTH=?>	(în pixeli)

Lungime procentuală	`<HR WIDTH="%">`	(procent din lățimea paginii)
Linie solidă	`<HR NOSHADE>`	(fără aspect 3D)
Linie text continuă	`<NOBR></NOBR>`	(previne trecerea la linie nouă)
Trecere la linie nouă	`<WBR>`	(unde e nevoie)

LISTE

Liste nenumerotate	`<UL>`
	`<LI>...`
	`</UL>`
		(<LI> înainte de fiecare element al listei)

Listă compactă	`<UL COMPACT></UL>`	(nenumerotată)
Tipul bullet-ului	`<UL TYPE=DISC|CIRCLE|SQUARE>`	(pentru toată lista)
	`<LI TYPE=DISC|CIRCLE|SQUARE>`	(acesta şi următorii)

Liste numerotate	`<OL>`
	`<LI>...`
	`</OL>`
		(<LI> înainte de fiecare element al listei)

Listă compactă	`<OL COMPACT></OL>`	(numerotată)
Tipul numărătorii	`<OL TYPE=A|a|I|i|1>`	(pentru toată lista)
	`<LI TYPE=A|a|I|i|1>`	(acesta şi subsecvențele sale)
Număr de început	`<OL START=?>`	(pentru toată lista)
	`<LI VALUE=?>`	(acesta şi subsecvențele sale)
Liste de definiții	`<DL><DT><DD></DL>`	(DT: termen DD: definiție)
Listă compactă	`<DL COMPACT></DL>`	(de definiții)

Liste de tip meniu	<MENU> <LI>... </MENU>	(<LI> înainte de fiecare element al listei)
Listă compactă	<MENU COMPACT></MENU>	(de meniu)
Liste de foldere	<DIR> <LI>... </DIR>	(<LI> înainte de fiecare element al listei)
Listă compactă	<DIR COMPACT></DIR>	(de foldere)

FUNDALURI ŞI CULORI

Imagine fundal	<BODY BACKGROUND="URL">	
Culoare fundal	<BODY BGCOLOR="#$$$$$$">	(ordinea este roşu/verde/albastru)
Culoare text	<BODY TEXT="#$$$$$$">	
Culoare link-uri	<BODY LINK="#$$$$$$">	
Link-uri vizitate	<BODY VLINK="#$$$$$$">	
Link activat	<BODY ALINK="#$$$$$$">	

CARACTERE SPECIALE (trebuie să fie minuscule)

Caracter special	&#?;	(? este cod ISO 8859-1)
<	<	
>	>	
&	&	
"	"	
Marcă înregistrată	®	
Copyright	©	

Spațiu neignorat	` `	

FORME

Definirea formei	`<FORM ACTION="URL" METHOD=GET|POST></FORM>`	
Încărcare fişier	`<FORM ENCTYPE="multipart/ form-data"> </FORM>`	
Câmp de introducere	`<INPUT TYPE="TEXT|PASSWORD| CHECKBOX|RADIO|IMAGE|HIDDEN| SUBMIT|RESET">`	
Numele câmpului	`<INPUT NAME="***">`	
Valoarea câmpului	`<INPUT VALUE="***">`	
Apăsat	`<INPUT CHECKED>`	(checkbox şi radio)
Dimensiunea câmpuluii	`<INPUT SIZE=?>`	(în caractere)
Dimensiune maximă	`<INPUT MAXLENGTH=?>`	(în caractere)
Listă de selecție	`<SELECT></SELECT>`	
Numele listei	`<SELECT NAME="***"></SELECT>`	
Numărul de opțiuni	`<SELECT SIZE=?></SELECT>`	
Opțiuni multiple	`<SELECT MULTIPLE>`	(se pot selecta mai multe)
Opțiune	`<OPTION>`	(elemente ce pot fi selectate)
Opțiune implicită	`<OPTION SELECTED>`	
Dimensiunea ferestrei text	`<TEXTAREA ROWS=? COLS=?> </TEXTAREA>`	
Numele ferestrei	`<TEXTAREA NAME="***"> </TEXTAREA>`	

Spațiere text	`<TEXTAREA WRAP=OFF|VIRTUAL|PHYSICAL> </TEXTAREA>`	

TABELE

Definire tabel	`<TABLE></TABLE>`	
Chenar tabel	`<TABLE BORDER=?></TABLE>`	
Spațiu între celule	`<TABLE CELLSPACING=?>`	
Spațiu față de marginea celulei	`<TABLE CELLPADDING=?>`	
Dimensiune dorită	`<TABLE WIDTH=?>`	(în pixeli)
Dimensiune procentuală	`<TABLE WIDTH="%">`	(procentaj din pagină)
Linie de tabel	`<TR></TR>`	
Aliniere	`<TR ALIGN=LEFT|RIGHT| CENTER|MIDDLE|BOTTOM VALIGN=TOP|BOTTOM|MIDDLE>`	
Celulă de tabel	`<TD></TD>`	(apare în interiorul unei linii de tabel)
Aliniere	`<TD ALIGN=LEFT|RIGHT| CENTER|MIDDLE|BOTTOM VALIGN=TOP|BOTTOM|MIDDLE>`	
Fără linii noi	`<TD NOWRAP>`	
Pe câte coloane se întinde	`<TD COLSPAN=?>`	
Pe câte linii se întinde	`<TD ROWSPAN=?>`	
Dimensiune dorită	`<TD WIDTH=?>`	(în pixeli)
Dimensiune procentuală	`<TD WIDTH="%">`	(procentaj din tabel)
Culoarea fundalului celulei	`<TD BGCOLOR="#$$$$$$">`	

Cap de tabel	`<TH></TH>`	(apare ca datele din interior, dar îngroşat,centrat)
Aliniere	`<TH ALIGN=LEFT|RIGHT| CENTER|MIDDLE|BOTTOM VALIGN=TOP|BOTTOM|MIDDLE>`	
Fără linii noi	`<TH NOWRAP>`	
Pe câte coloane se întinde	`<TH COLSPAN=?>`	
Pe câte linii se întinde	`<TH ROWSPAN=?>`	
Dimensiune dorită	`<TH WIDTH=?>`	(în pixeli)
Dimensiune procentuală	`<TH WIDTH="%">`	(procentaj din tabel)
Culoare fundal	`<TH BGCOLOR="#$$$$$$">`	
Captură tabel	`<CAPTION></CAPTION>`	
Aliniere	`<CAPTION ALIGN=TOP|BOTTOM>`	(înainte/după tabel)

FERESTRE

Ferestre document	`<FRAMESET> </FRAMESET>`	(în loc de <BODY>)
Înălțimea liniilor	`<FRAMESET ROWS=,,,> </FRAMESET>`	(pixeli sau %)
Înălțimea liniilor	`<FRAMESET ROWS=*> </FRAMESET>`	(*: dimensiune relativă)
Lățimea coloanelor	`<FRAMESET COLS=,,,> </FRAMESET>`	(pixeli sau %)
Lățimea coloanelor	`<FRAMESET COLS=*> </FRAMESET>`	(*: dimensiune relativă)
Dimensiunea marginilor	`<FRAMESET BORDER=?>`	
Margine	`<FRAMESET FRAMEBORDER="yes|no">`	

Culoarea marginii	`<FRAMESET BORDERCOLOR="#$$$$$$">`	
Definire fereastră	`<FRAME>`	
URL afişat în fereastră	`<FRAME SRC="URL">`	
Numele ferestrei	`<FRAME NAME="***|_blank| _self|_parent|_top>`	
Dimensiunea marginii	`<FRAME MARGINWIDTH=?>`	(margine stânga şi dreapta)
Dimensiunea marginii	`<FRAME MARGINHEIGHT=?>`	(marginea de sus şi de jos)
Bară de defilare	`<FRAME SCROLLING="YES|NO|AUTO">`	
Neredimensionabil	`<FRAME NORESIZE>`	
Margini	`<FRAME FRAMEBORDER="yes|no">`	
Culoarea marginii	`<FRAME BORDERCOLOR="#$$$$$$">`	
Conținut fără ferestre	`<NOFRAMES> </NOFRAMES>`	(pt. browser-e ce nu cunosc cadre

JAVA

Applet	`<APPLET></APPLET>`	
Nume applet	`<APPLET CODE="***">`	
Parametri	`<APPLET PARAM NAME="***">`	
Locație applet	`<APPLET CODEBASE="URL">`	
Identificator applet	`<APPLET NAME="***">`	(pentru referiri din altă parte a paginii)
Text alternativ	`<APPLET ALT="***">`	(pentru browser-e non-Java)

Aliniere	`<APPLET ALIGN="LEFT|RIGHT|CENTER">`	
Dimensiune	`<APPLET WIDTH=? HEIGHT=?>`	(în pixeli)
Distanțare	`<APPLET HSPACE=? VSPACE=?>`	(în pixeli)

DIVERSE

Comentariu	`<!-- *** -->`	(neafişat de browser)
HTML 3.2 Prolog	`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">`	
De căutat	`<ISINDEX>`	(indică un index de căutat)
Afişare	`<ISINDEX PROMPT="***">`	(text de afişat la introducere)
Trimitere căutare	`<A HREF="URL?***"> </A>`	(se foloseşte un semn de întrebare)
URL fişier curent	`<BASE HREF="URL">`	(trebuie să fie în header)
Nume de bază fereastră	`<BASE TARGET="***">`	(trebuie să fie în header)
Relație	`<LINK REV="***" REL="***" HREF="URL">`	(trebuie să fie în header)
Meta-informații	`<META>`	(trebuie să fie în header)
Foi de stil	`<STYLE></STYLE>`	
Script-uri JavaScript	`<SCRIPT></SCRIPT>`	

ANEXA 2: O APLICAȚIE C++ PENTRU CREAREA SCRIPT-URILOR CGI

Fișierul header `parse.h` (declarația clasei `Parse`)

```
// Extrage subșirurile components ale unui rezultat CGI
// Exemplu:
// name=Your+name&action=%2B10%25&log=~mas/log
// este compus din trei elemente:
//      Element: șirul asociat elementului
//      name: Your name
//      action: +10%
//      log: /usr/staff/mas/log
// (C) M.A.Smith University of Brighton/
// Permission is granted to use this code/
// provided this declaration and copyright notice remains intact./
// 21 October 1995
#ifndef CLASS_PARSE
#define CLASS_PARSE

#include "t_type.h"

class Parse
{
public:
  Parse( char [] );
  ~Parse();
  void set( char [] );
  char *get_item( char [], int pos=1, bool=false );
  char *get_item_n( char [], int pos=1, bool=false );
protected:
  void remove_escape(char []);    // Strip escape characters
  int hex( char );                // Return hex value
  char *map_uname( char [] );     // process ~uname -> home path
private:
  enum { SEP = '&' };             // Seperator for fields
  char *the_str;                  // String parsed
  int  the_length;                // Length of the string
};

#endif
```

Fișierul sursă `parse.cpp` (expandarea metodelor clasei `Parse`)

```
// (C) M.A.Smith University of Brighton
// Permission is granted to use this code
//   provided this declaration and copyright notice remains intact.
// 21 October 1995

#ifndef CLASS_PARSE_IMP
#define CLASS_PARSE_IMP

#include "parse.h"

#include <string.h>
#include <ctype.h>
#ifndef NO_MAP
# include <pwd.h>
#endif

Parse::Parse( char list[] )
{
  the_str = NULL;
  set( list );}
```

```
Parse::~Parse()
{
  if ( the_str != NULL ) {          // Release storage
    delete [] the_str;
  }
}

void Parse::set( char list[] )
{
  if ( the_str != NULL ) {          // Release storage
    delete [] the_str;
  }
  the_length = strlen( list );      // Length of string
  the_str = new char[the_length+1]; // Allocate area
  strcpy( the_str, list );          // copy to
}

char *Parse::get_item( char name[], int pos, bool file )
{
  int len = strlen( name );
  int cur_tag = 1;
  for( int i=0; i<the_length-len; i++ )
  {
    if ( the_str[i] == name[0]  &&
         strncmp( &the_str[i], name, len ) == 0 )
    {
      if ( the_str[i+len] == '=' )
      {
        if ( cur_tag == pos )
        {
          int start = i+len+1; int j = start;
          while ( the_str[j] != SEP && the_str[j] != '\0' ) j++;
          int str_len = j-start;
          char *mes = new char[ str_len+1 ];
          strncpy( mes, &the_str[start], str_len );
          mes[str_len] = '\0';
          remove_escape( mes );
#         ifndef NO_MAP
          return file ? map_uname(mes) : mes;
#         else
          return file ? mes : mes;
#         endif
        } else {
          cur_tag++;
        }
      }
    }
  }
  return NULL;
}

char *Parse::get_item_n( char name[], int pos, bool file )
{
  char *res = get_item( name, pos, file );
  return res == NULL ? "----" : res;
}

void Parse::remove_escape(char str[])
{
  char * from = &str[0];
  char * to   = &str[0];
  while ( *from != '\0' )
  {
    char ch = *from++;
    switch ( ch )
    {
      case '%' :
        ch = (hex(*from++)<<4) | hex(*from++);
        break;
      case '+' :
        ch = ' '; break;
```

```
    }
    *to++ = ch;
  }
  *to = '\0';
}

int Parse::hex( char c )
{
  if ( isdigit( c ) )
    return c-'0';
  if ( isalpha( c ) )
    return tolower(c)-'a'+10;
  return 0;
}

char* Parse::map_uname( char *path )
{
#ifndef NO_MAP
  if ( path[0] == '~' )
  {
     char uname[255];                        // Holds user name
     char *rest = &path[1];                  // omit ~
     char *p = &uname[0];                    // user

     while ( *rest != '/' && *rest != '\0' )
     {
       *p++ = *rest++;
     }
     *p = '\0';                              // Terminator
     char *root = uname;                     // If fails
     passwd *pw = getpwnam( uname );         // Structure about user
     if ( pw != NULL )
     {
       root = pw->pw_dir;                    // Users home directory
     }
     int len_root = strlen(root);
     int len_path  = len_root + strlen(rest);
     char *new_path = new char[len_path+1]; // Dynamic string

     strcpy( &new_path[0], root );           // abs user path
     strcpy( &new_path[len_root], rest );    // remainder
     return new_path;
  } else {
    return path;
  }
#endif
 return path;
}
#endif
```

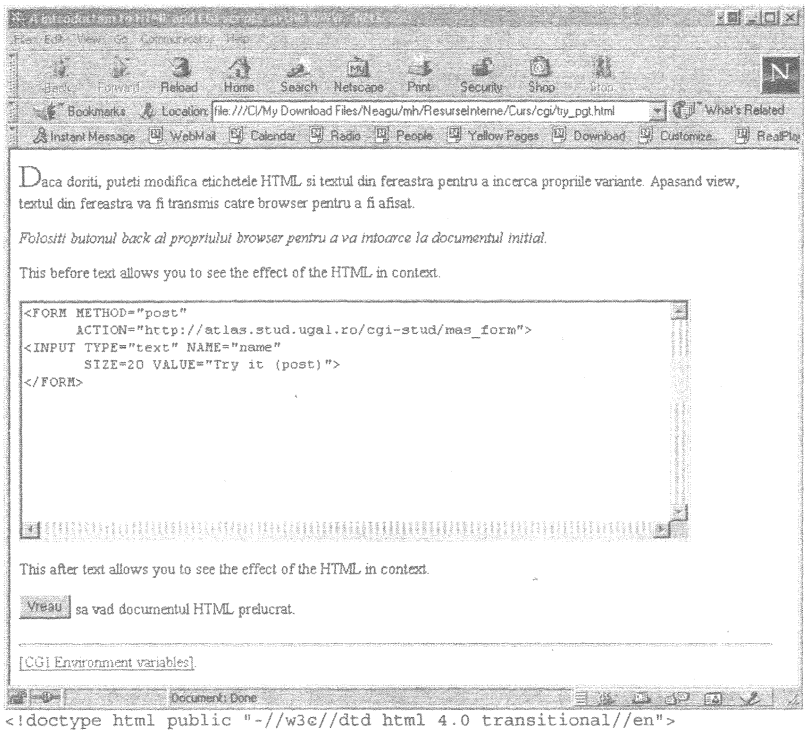
Fişierul header `t_type.h` (declarații ale unor constante şi clase ajutătoare pentru parsing):

```
#ifndef MAS_TYPE
#define MAS_TYPE

#ifndef __TURBOC__
# define __TURBOC__ 0x0000
#endif

#define V5 (__TURBOC__ >= 0x0500 )

#define V4 (__TURBOC__ >= 0x0400 && __TURBOC__ <= 0x0499 )

#define V3 (__TURBOC__ >= 0x0300 && __TURBOC__ <= 0x0399 )

#if V3
#define DOES_NOT_HAVE_BOOL
#define DOES_NOT_HAVE_STRING
#define DOES_NOT_HAVE_EXCEPTION
```

```
#define DOES_NOT_HAVE_EXCEPTION_CLASSES
#define DOES_NOT_HAVE_MUTABLE
#define DOES_NOT_HAVE_EXPLICIT
#define DOES_NOT_HAVE_NAMESPACE_STD
#endif

#if V4
#define DOES_NOT_HAVE_BOOL
#define DOES_NOT_HAVE_EXCEPTION_CLASSES
#define DOES_NOT_HAVE_MUTABLE
#define DOES_NOT_HAVE_EXPLICIT
#define DOES_NOT_HAVE_NAMESPACE_STD
#endif

#if V5
#define DOES_NOT_HAVE_EXCEPTION_CLASSES
//#define HAS_NAMESPACE_STD
#endif

#ifdef __GNUC__
//#define NEED_TO_INCLUDE_libioP
//#define DOES_NOT_HAVE_EXCEPTION -fhandle-execeptions
#define DOES_NOT_HAVE_EXCEPTION_CLASSES
//#include <std/stdexcep.h> (Broken)
#define DOES_NOT_HAVE_MUTABLE
#define DOES_NOT_HAVE_EXPLICIT
#define DOES_NOT_HAVE_NAMESPACE_STD
#endif

#ifdef DOES_NOT_HAVE_NAMESPACE_STD
#define std
#endif

#ifdef NEED_TO_INCLUDE_libioP
#include <libioP.h>
#endif

#ifdef DOES_NOT_HAVE_BOOL
# define bool int
# define true 1
# define false 0
#endif

#ifdef DOES_NOT_HAVE_MUTABLE
# define mutable
#endif

#ifdef DOES_NOT_HAVE_EXPLICIT
# define explicit
#endif

#ifdef DOES_NOT_HAVE_EXCEPTION
# define throw
# define try
# define catch( parameter ) exception err; if ( false )
#endif

#undef DOES_NOT_HAVE_EXCEPTION_CLASSES
#ifdef DOES_NOT_HAVE_EXCEPTION_CLASSES

# ifdef DOES_NOT_HAVE_STRING
#   include "t99_str.h"
# else
#   ifdef __GNUC__
#     include <string.h>
#   else
#     include <cstring.h>
#   endif
# endif
```

```
class exception
{
public:
  exception( const string& arg="" )
  {
    the_message = arg;
  }
  virtual ~exception(){};
  virtual const char* what()
  {
    return the_message.c_str();
  }
private:
  string the_message;
};

class logic_error: public exception
{
public:
  logic_error( const string& arg ) : exception( arg ) {};
};

class runtime_error: public exception
{
public:
  runtime_error( const string& arg ) : exception( arg ) {};
};

class range_error: public runtime_error
{
public:
  range_error( const string& arg ) : runtime_error( arg ) {};
};

class overflow_error: public runtime_error
{
public:
  overflow_error( const string& arg ) : runtime_error( arg ) {};
};

class domain_error: public logic_error
{
public:
  domain_error( const string& arg ) : logic_error( arg ) {};
};

class out_of_range: public logic_error
{
public:
  out_of_range( const string& arg ) : logic_error( arg ) {};
};

class invalid_argument: public logic_error
{
public:
  invalid_argument( const string& arg ) : logic_error( arg ) {};
};

class length_error: public logic_error
{
public:
  length_error( const string& arg ) : logic_error( arg ) {};
};
#endif
#endif
```

Un exemplu de utilizare a clasei `Parse` pentru interpretarea unui script CGI (v. figura). Fişierul gazdă a apelului de script CGI, `try_pgt.html`:




```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
   <meta name="KEYWORDS" content="HTML, HTML CGI">
   <meta name="DESCRIPTION" content="HTML WWW CGI">
   <title>A introduction to HTML and CGI scripts on the WWW</title>
<!-- ------------------------------------------------- -->
</head>
<body bgcolor="#FFFFFF" link="#008080" vlink="#008080" alink="#808000">
<font size=+3>D</font>aca
doriti, puteti modifica etichetele HTML si textul din fereastra pentru
a incerca propriile variante. Apasand view, textul din fereastra va fi
transmis catre browser pentru a fi afisat.
<p><i>Folositi butonul back al propriului browser pentru a va intoarce
la documentul initial.</i>
<P>
This before text allows you to see the effect of the HTML in context.

<FORM ACTION="http://atlas.stud.ugal.ro/cgi-stud/mas_try">
 <TEXTAREA NAME="feedback" ROWS=12 COLS=75>
&lt;FORM METHOD="post"
      ACTION="http://atlas.stud.ugal.ro/cgi-stud/mas_form">
&lt;INPUT TYPE="text" NAME="name"
       SIZE=20 VALUE="Try it (post)">
&lt;/FORM>
 </TEXTAREA>
 <P>
 This after text allows you to see the effect of the HTML in context.
 <P>
 <INPUT TYPE="submit" NAME="button" VALUE="Vreau"> sa vad documentul HTML
prelucrat.
</FORM>

<HR>
```

```
<a href=
"http://atlas.stud.ugal.ro/cgi-
stud/mas_rec?page=mas&file=~pctp/mas_logs">
[CGI Environment variables]</a><img SRC="mas_rec" ALT="CGI" height=1
width=1>
</body>
</html>
```

Fişierul sursă `mas_try.cpp`:

```
// (C) M.A.Smith University of Brighton
// Permission is granted to use this code
//   provided this declaration and copyright notice remains intact.
// 15 September 1995

#include "t_type.h"
#define NO_MAP

#include "iostream.h"
#include <stdlib.h>
#include <string.h>

#define SERVER  "http://atlas.stud.ugal.ro/"
#define TRY_PROG "cgi-stud/mas_try"

#include "parse.h"
#include "parse.cpp"

void process_script( char [], bool );
inline void html( char str[] ) { cout << str << "\n"; }
inline void html_( char str[] ) { cout << str; }
inline void html_( char c ) { cout << c; }

// Main program
int main()
{
  char *query_str = getenv("QUERY_STRING");
  Parse list( query_str == 0 ? "feedback=xxx&nowrap" : query_str );

  process_script( list.get_item( "feedback" ),
                  list.get_item("nowrap")==NULL );
  return 0;
}

void process_script( char mes[], bool wrap )
{
  html( "Content-type: text/html" );
  html( "" ); html( "" );
  if( wrap )
  {
    html("<HTML>");
    html("<HEAD>");
    html("<TITLE>Example of HTML</TITLE>");
    html("</HEAD>");
    html("<BODY BGCOLOR=White>");
    html("<H2>Result of processing the HTML</H2>");
    html("<P>");
    html("This before text allows you to see the effect "
         "of the HTML in context.");
  }
  html( mes );
  if ( wrap )
  {
    html("This after text allows you to see the effect "
         "of the HTML in context.");

    html("</BODY>");
    html("</HTML>");
  }
}
```

Fişierul sursă `mas_form.cpp`:

```
// (C) M.A.Smith University of Brighton
// Permission is granted to use this code
//   provided this declaration and copyright notice remains intact.
// 26 August 1995

#include "t_type.h"
#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>
#include "mas_cvo.cpp"

void intro();
void finish();
void form_data_output();
void stream_data();
void return_button();
char* getenv_n( char [] );

// Main program
int main()
{
  intro();
  form_data_output();
  stream_data();
  return_button();
  finish();
  return 0;
}

void intro()
{
  cout << "Content-type: text/html" << "\n"
       << "\n" << "\n" ;
  cout << "<HTML>  " << "\n";
  cout << "<HEAD>  " << "\n";
  cout << "</HEAD> " << "\n";
  cout << "<BODY BGCOLOR=White>  " << "\n";
}

void form_data_output( )
{
  cout << "<P>" << "\n";
  cout << "The data sent to the form processing program " <<
          "in the environment variable QUERY_STRING is:" << "\n";
  cout << "<P>" << "\n" << "\n";
  cout << "<TABLE BORDER CELLPADDING=2>" << "\n";
  cout << "<TD>" << "\n";
  cout << getenv_n( "QUERY_STRING" ) <<  "\n";
  cout << "</TD>" << "\n";
  cout << "</TABLE>" << "\n";
}

void stream_data()
{
  char ch;
  cout << "<P>" << "\n";
  cout << "Contents of stdin" << "<P>" << "\n";
  cout << "<TABLE BORDER CELLPADDING=2>" << "\n";
  cout << "<TD>" << "\n";
  cout << "<PRE>" << "\n";
  cout << "" << "\n";
  {
    int len = 0;
    char *p_ch = getenv( "CONTENT_LENGTH" );
    if ( p_ch != NULL ) len = atoi( p_ch );
    if ( len > 0 )
    {
      cin >> resetiosflags( ios::skipws );
```

```
      while ( cin >> ch, !cin.eof() )
      {
        cout << ch;
        if ( --len <=0 ) break;
      }
    }
  }
  cout << "</PRE>" << "\n";
  cout << "</TD>" << "\n";
  cout << "</TABLE>" << "\n";
}

void return_button()
{
  cout << "<P>" << "\n";
  cout << "<TABLE BORDER CELLPADDING=2>" << "\n";
  cout << "<TD>" << "\n";
  cout << "<A HREF=\"" << getenv_n( "HTTP_REFERER" ) << "\" >" <<
          "Return" << "</A>" << "\n";
  cout << "</TD>" << "\n";
  cout << "</TABLE>" << "\n";
}

void finish()
{
  cout << "</BODY>" << "\n";
  cout << "</HTML>" << "\n";
}
```

Fişierul sursă `mas_cvo.cpp`:

```
#include "t_type.h"
#include <iostream.h>
#include <stdlib.h>

inline void html( char str[] ) { cout << str << "\n"; }
inline void html_( char str[] ) { cout << str; }

inline void html_( char c ) { cout << c; }

// Debug option to deliver
// environment variables seen by the script

char* getenv_n( char var[] )
{
  char *p = getenv( var );
  return p == NULL ? "[]" : p;
}

void cgi_var_output( )
{
  html("<HTML>");
  html("<BODY BGCOLOR=White>");
  html( "<PRE>" );
  html_("AUTH_TYPE          "); html( getenv_n( "AUTH_TYPE" ) );
  html_("CONTENT_LENGTH     "); html( getenv_n( "CONTENT_LENGTH" ) );
  html_("CONTENT_TYPE       "); html( getenv_n( "CONTENT_TYPE" ) );
  html_("GATEWAY_INTERFACE  "); html( getenv_n( "GATEWAY_INTERFACE" ) );
  html_("HOSTTYPE           "); html( getenv_n( "HOSTTYPE" ) );
  html_("HTTP_ACCEPT        "); html( getenv_n( "HTTP_ACCEPT" ) );
  html_("HTTP_CONNECTION    "); html( getenv_n( "HTTP_CONNECTION" ) );
  html_("HTTP_HOST          "); html( getenv_n( "HTTP_HOST" ) );
  html_("HTTP_PRAGMA        "); html( getenv_n( "HTTP_PRAGMA" ) );
  html_("HTTP_REFERER       "); html( getenv_n( "HTTP_REFERER" ) );
  html_("HTTP_USER_AGENT    "); html( getenv_n( "HTTP_USER_AGENT" ) );
  html_("OSTYPE             "); html( getenv_n( "OSTYPE" ) );
  html_("PATH_INFO          "); html( getenv_n( "PATH_INFO" ) );
  html_("PATH_TRANSLATED    "); html( getenv_n( "PATH_TRANSLATED" ) );
  html_("QUERY_STRING       "); html( getenv_n( "QUERY_STRING" ) );
  html_("REMOTE_ADDR        "); html( getenv_n( "REMOTE_ADDR" ) );
  html_("REMOTE_HOST        "); html( getenv_n( "REMOTE_HOST" ) );
```

```
  html_("REMOTE_IDENT      "); html( getenv_n( "REMOTE_IDENT" ) );
  html_("REMOTE_USER       "); html( getenv_n( "REMOTE_USER" ) );
  html_("REQUEST_METHOD    "); html( getenv_n( "REQUEST_METHOD" ) );
  html_("SCRIPT_NAME       "); html( getenv_n( "SCRIPT_NAME" ) );
  html_("SERVER_NAME       "); html( getenv_n( "SERVER_NAME" ) );
  html_("SERVER_PORT       "); html( getenv_n( "SERVER_PORT" ) );
  html_("SERVER_PROTOCOL   "); html( getenv_n( "SERVER_PROTOCOL" ) );
  html_("SERVER_SOFTWARE   "); html( getenv_n( "SERVER_SOFTWARE" ) );

  html("<P>");
  html("Unix command printenv");
  cout << flush;
  system( "printenv | sort" );

  html("<P>");
  html("Unix command set");
  cout << flush;
  system( "set | sort " );

  html( "</PRE>" );
  html( "</BODY>" );
  html( "</HTML>" );
}
```

Fişierul sursă `mas_rec.cpp`:

```
// Record details about who the viewer of a web page is
// (C) M.A.Smith University of Brighton
// Permission is granted to use this code
//   provided this declaration and copyright notice remains intact.
// 26 August 1995

#include "t_type.h"
#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <ctype.h>
#include "parse.h"
#include "parse.cpp"
#include "mas_cvo.cpp"

void write_log_entry( char[], char[] );
void cgi_var_output();
char* getenv_n( char [] );
void gif_output( char [] );
// Parameters to the CGI program
//  passed in the QUERY_STRING environment variable
//
//  file=mame   - Names the file to which log information is written
//  page=name   - The name of the page that will be written to the log
//  img=name    - Names the image file which is returned
//              - If no name or unreadable an image of a . is returned
//              - If this parameter is omitted then a web page of CGI
//              -  variables is returned to help in debuging
// Remember all files name must be absolute or relative to the
//  directory in which the CGI script is run
int main()
{
  char *query_str = getenv("QUERY_STRING");
  Parse list( query_str == 0 ?
            "file=mas&page=Test&" : query_str );
  if ( list.get_item( "file" ) != NULL )
  {
    write_log_entry( list.get_item_n( "file", 1, true ),
                     list.get_item_n( "page" ) );
  }
  if ( list.get_item( "img" ) != NULL )
  {
```

```
    gif_output( list.get_item_n("img", 1, true) );
  } else {
    html("Content-type: text/html"); html("");
    cgi_var_output();       // debug option
  }
  return 0;
}
void write_log_entry( char file[], char page[] )
{
  //cout << "File = [" << file << "] page = [" << page << "]" << "\n";
  //cout.flush();
  ofstream inf( file, ios::app );
  //inf.open( file,  ios::out );
  if ( !inf.fail() )
  {
    time_t t;
    time( &t);
    char *str = ctime( &t ); str[24] = '\0';
    inf << setiosflags( ios::left );
    inf << setw(24) << str << " " <<
          setw(10) << (page!=NULL? page : "Unknown" ) << " " <<
          setw(18) << getenv_n("REMOTE_ADDR") << " " <<
          setw(20) << getenv_n("REMOTE_HOST") << "\n";
    //    setw(20) << getenv_n("REMOTE_USER") << "\n";
  }
}

void gif_output( char gif[] )
{
  char square [] = {
    'G',  'I',  'F',  '8',  '9',  'a',
    0002, 0000, 0002, 0000, 0263, 0000, 0000, 0000, 0000, 0000,
    0277, 0000, 0000, 0000, 0277, 0000, 0277, 0277,
    0000, 0000, 0000, 0277, 0277, 0000, 0277, 0000,
    0277, 0277, 0300, 0300, 0300, 0200, 0200, 0200,
    0377, 0000, 0000, 0000, 0377, 0000, 0377, 0377,
    0000, 0000, 0000, 0377, 0377, 0000, 0377, 0000,
    0377, 0377, 0377, 0377, 0377, 0054, 0000, 0000,
    0000, 0000, 0002, 0000, 0002, 0000, 0100, 0004,
    0003, 0020, 0200, 0010, 0000, 0073 };
  html("Content-type: image/gif"); html("");
  if ( gif[0] != '\0' )              // Output gif file
  {
    ifstream in( gif );
    if ( !in.fail() )                // Can read
    {
      char c;
      in.read( &c, 1 );
      while ( !in.eof() )
      {
        cout.write( &c, 1 );
        in.read( &c, 1 );
      }
      return;                        // Finish
    }
  }
  cout.write( square, (int) sizeof( square ) );
}
```

BIBLIOGRAFIE

ACM SIGWEB Ninth International **World Wide Web Conference** proceedings, Amsterdam, May 2001.

ACM Transactions on Internet Technology, vol. I, Aug. 2001.

I. Athanasiu, B. Costinescu, O.A. Drăgoi, F.I. Popovici, **Limbajul Java. O perspectivă pragmatică**. Agora, Tg.Mureş, 1998.

J.G. Brookshear, **Introducere in informatică**, Editura Teora, 1999.

N. Chapman, J. Chapman, **Digital Multimedia**, ISBN: 0471983861, John Wiley and Sons, 2000.

Corel Corporation documentation:
http://www.corel.com/products/CaseStudies/coreldraw_casestudies.htm

R. Darnel, **Totul despre HTML 4**, Editura Teora, 2001.

C. Fraizer, J. Bond, **Java API. Pachetele API java.applet si java awt**. Teora, Bucureşti, 1998.

IBM Learning Services, **Building a Java applet**, April 1999, http://www-105.ibm.com/developerworks/education.nsf/java-onlinecourse-bytitle/

IEEE Computer Graphics and Applications Magazine, 2000 and 2001 issues, IEEE 016-306-TBR.

IEEE Trans. on Multimedia, 2000 and 2001 issues, IEEE 500-182-TBR.

T. Ionescu, **Sistemul WINDOWS**, Editura MatrixRom, 2001.

J. Kraynak, **Internet in imagini**, Editura Teora, 2000.

D. Lowe, W. Hall, **Hypermedia and the Web**, John Willey & sons 1999.

R. Maran, P. Whitehead, **Computers Simplified: Simply the Easiest Way to Learn**, Marangraphics Inc. 2001.

C. Masalagiu, I. Asiminoaei, **Metodica predării informaticii**, Editura MatrixRom, 2001.

Microsoft Corporation, **Office 2000 Resources for Educators**:
http://www.microsoft.com/education

C.D. Neagu, D. Neagu, **Pachete de aplicaţii pentru birotică şi desen**, Editura MatrixRom, 2001.

C.D. Neagu, S. Bumbaru, **Sisteme multimedia. Grafică pe calculator**, Editura MatrixRom, 2001.

L.A. Phillips, **XML**, Editura Teora, 2000.

E. Rotariu, **Limbajul Java**. Agora, Tg.Mureş, 1996.

N. Snell, **Crearea paginilor Web**, Editura Teora, 2000.

Sun, **JAVA: programming language and env.** http://java.sun.com/

World Wide Web Consortium (W3C): http://www.w3.org/

WebGuest Web Glossary: http://www.webguest.com/glossary

J.K. Wu, M.S. Kankanhalli, J.H. Lim, D. Hong, **Perspectives on Content-Based Multimedia Systems**, Kluwer, 2000.

UNIVERSITATEA
BIBLIOTECA 2

CĂRȚI TEHNICE - MATRIX ROM - CĂRȚI ACADEMICE

9 ani de activitate - peste 900 de titluri

În domeniul **CALCULATOARE*** au mai apărut:

- Sabin Corneliu Buraga (Universitatea Al.I.Cuza Iaşi) - **TEHNOLOGII WEB** • Preț: 215000 lei
- Mihai Ceapăru (Universitatea "Politehnica" Bucureşti) - **PROGRAMMATION EN LANGAGE D'ASSEMBLAGE** • Preț: 90000 lei
- Francis Cottet, Octavian Ciobanu (Universitatea tehnică "Gh.Asachi" Iaşi) - **BAZELE PROGRAMĂRII ÎN LABVIEW** • Preț: 80000 lei
- S. Curteanu (Universitatea tehnică "Gh.Asachi" Iaşi) - **CALCUL NUMERIC ŞI SIMBOLIC ÎN MATHCAD** • Preț: 95000 lei
- Tone Ionescu (Universitatea tehnică de construcții Bucureşti) - **SISTEMUL WINDOWS** • Preț: 80000
- Toader Jucan (Universitatea Al.I.Cuza Iaşi) - **LIMBAJE FORMALE ŞI AUTOMATE** • Preț: 75000
- Liviu Kreindler, Răducu Giuclea (Universitatea "Politehnica" Bucureşti) - **BAZELE MICROPROCESOARELOR** • Preț: 99000 lei
- C.Masalagiu, I. Asiminoaei (Universitatea Al.I.Cuza Iaşi) - **METODICA PREDĂRII INFORMATICII** • Preț: 60000 lei
- Irina Munteanu, Daniel Ioan (Universitatea "Politehnica" Bucureşti) - **UTILIZAREA INSTRUMENTELOR UNIX ÎN ŞTIINȚĂ ŞI INGINERIE** • Preț: 120000 lei
- Ciprian Daniel Neagu, Severin Bumbaru (Universitatea Dunărea de Jos Galați) - **SISTEME MULTIMEDIA. GRAFICĂ PE CALCULATOR** • Preț: 75000 lei
- Ciprian Daniel Neagu, Daniela Neagu (Universitatea Dunărea de Jos Galați) - **PACHETE DE APLICAȚII PENTRU BIROTICĂ ŞI DESEN** • Preț: 65000 lei
- Ciprian Daniel Neagu, Daniela Neagu (Universitatea Dunărea de Jos Galați) - **HYPERMEDIA** •
- Eugen Petac, Dorina Petac (Universitatea "Ovidius" Constanța) - **PRINCIPII ŞI TEHNICI DE PROTECȚIE A INFORMAȚIEI ÎN REȚELELE DE CALCULATOARE** • Preț: 80000 lei
- Daniela Elena Popescu (Universitatea Oradea) - **INTRODUCERE ÎN ARHITECTURA SISTEMELOR CONVENȚIONALE DE CALCUL** • Preț: 125000 lei
- Costin Pribeanu (Institutul de Cercetări în Informatică) - **BAZE DE DATE ŞI APLICAȚII** • Preț: 99000 lei
- Costin Pribeanu (Institutul de Cercetări în Informatică) - **PROIECTAREA INTERFEȚELOR OM-CALCULATOR** • Preț: 72000 lei
- T. Prisecaru, A.S. Ene (Universitatea "Politehnica" Bucureşti) - **LIMBAJUL DE PROGRAMARE C++. NOȚIUNI DE BAZĂ** • Preț: 85000 lei
- Ştefan Trăuşan-Matu (Universitatea "Politehnica" Bucureşti) - **INTERFAȚAREA EVOLUATĂ OM-CALCULATOR** • Preț: 75000 lei
- Ştefan Trăuşan-Matu, Claudiu Răibuleț (Universitatea "Politehnica" Bucureşti) - **PRELUCRAREA DOCUMENTELOR FOLOSIND XML ŞI PERL** • Preț: 62000 lei

Ofertele complete şi gratuite pe domenii se pot solicita telefonic. Achiziționarea cărților se poate face direct de la sèdiul editurii, prin colet poştal cu plata ramburs (pe baza unei comenzi scrise) sau de la distribuitorii din Bucureşti (librăria RAMA, librăria 164, librăria AGIR, librăria Luceafărul, librăria Universității), Bacău, Braşov, Cluj Napoca, Constanța, Iaşi, Oradea, Reşița, Târgu Jiu. Clienții primesc periodic informații despre noile lucrări apărute sau în curs de apariție.

*** Prețurile din această listă sunt prețuri de vânzare directă, valabile începând cu 01.01.2002**